NETCONF Working Group
Internet-Draft
Obsoletes: 5539 (if approved)
Intended status: Standards Track
Expires: April 25, 2012

M. Badra
DU

October 23, 2011

NETCONF Over Transport Layer Security (TLS)
draft-badra-netconf-rfc5539bis-00

Abstract

The Network Configuration Protocol (NETCONF) provides mechanisms to install, manipulate, and delete the configuration of network devices. This document describes how to use the Transport Layer Security (TLS) protocol to secure NETCONF exchanges. This document obsoletes RFC 5539.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 25, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as

described in the Simplified BSD License.

Table of Contents

1. Introduction 3
1.1. Conventions Used in This Document 3
2. NETCONF over TLS 3
2.1. Connection Initiation 3
2.2. Connection Closure 4
3. Endpoint Authentication, Identification and Authorization . . 5
3.1. Server Identity 5
3.2. Client Identity 6
3.2.1. Deriving NETCONF Usernames From NETCONF Client Certificates 6
3.2.2. Deriving NETCONF Usernames From PSK identities 14
4. Security Considerations 15
5. IANA Considerations 15
6. Acknowledgements 16
7. Contributor’s Address 16
8. References 16
8.1. Normative References 16
8.2. Informative References 17
Appendix A. Change Log (to be removed by RFC Editor before publication) 17
A.1. From RFC5539 to draft-badra-netconf-rfc5539bis-00 17
Author’s Address 18

1. Introduction

The NETCONF protocol [RFC6241] defines a mechanism through which a network device can be managed. NETCONF is connection-oriented, requiring a persistent connection between peers. This connection must provide integrity, confidentiality, peer authentication, and reliable, sequenced data delivery.

This document defines "NETCONF over TLS", which includes support for certificate and pre-shared key (PSK)-based authentication and key derivation, utilizing the protected ciphersuite negotiation, mutual authentication, and key management capabilities of the TLS (Transport Layer Security) protocol, described in [RFC5246].

1.1. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. NETCONF over TLS

Since TLS is application-protocol-independent, NETCONF can operate on top of the TLS protocol transparently. This document defines how NETCONF can be used within a TLS session.

2.1. Connection Initiation

The peer acting as the NETCONF client MUST also act as the TLS client. It MUST connect to the server that passively listens for the incoming TLS connection on the TCP port 6513. It MUST therefore send the TLS ClientHello message to begin the TLS handshake. Once the TLS handshake has finished, the client and the server MAY begin to exchange NETCONF data. In particular, the client will send complete XML documents to the server containing <rpc> elements, and the server will respond with complete XML documents containing <rpc-reply> elements. The client MAY indicate interest in receiving event notifications from a server by creating a subscription to receive event notifications [RFC5277]. In this case, the server replies to indicate whether the subscription request was successful and, if it was successful, the server begins sending the event notifications to the client as the events occur within the system.

All NETCONF messages MUST be sent as TLS "application data". It is possible that multiple NETCONF messages be contained in one TLS record, or that a NETCONF message be transferred in multiple TLS records.

The previous version [RFC5539] of this document uses the same delimiter sequence defined in [RFC4742], under the assumption that it could not be found in well-formed XML documents. However, this assumption is not correct [RFC6242]. In order to solve this problem, and at the same time be compatible with existing implementations, this document uses the framing protocol defined in [RFC6242] as following :

The <hello> message MUST be followed by the character sequence]]>]]>. Upon reception of the <hello> message, the receiving peer's TLS Transport layer conceptually passes the <hello> message to the Messages layer. If the :base:1.1 capability is advertised by both peers, the chunked framing mechanism defined in Section 4.2 of [RFC6242] is used for the remainder of the NETCONF session. Otherwise, the old end-of-message-based mechanism (see Section 4.3 of [RFC6242]) is used.

Implementation of the protocol specified in this document MAY implement any TLS cipher suite that provides mutual authentication [RFC5246].

Implementations MUST support TLS 1.2 [RFC5246] and are REQUIRED to support the mandatory-to-implement cipher suite, which is TLS_RSA_WITH_AES_128_CBC_SHA. This document is assumed to apply to future versions of TLS; in which case, the mandatory-to-implement cipher suite for the implemented version MUST be supported.

2.2. Connection Closure

A TLS client MUST close the associated TLS connection if the connection is not expected to issue any NETCONF RPC commands later. It MUST send a TLS close_notify alert before closing the connection. The TLS client MAY choose to not wait for the TLS server close_notify alert and simply close the connection, thus generating an incomplete close on the TLS server side. Once the TLS server gets a close_notify from the TLS client, it MUST reply with a close_notify unless it becomes aware that the connection has already been closed by the TLS client (e.g., the closure was indicated by TCP).

When no data is received from a connection for a long time (where the application decides what "long" means), a NETCONF peer MAY close the connection. The NETCONF peer MUST attempt to initiate an exchange of close_notify alerts with the other NETCONF peer before closing the connection. The close_notify's sender that is unprepared to receive any more data MAY close the connection after sending the close_notify alert, thus generating an incomplete close on the close_notify's receiver side.

3. Endpoint Authentication, Identification and Authorization

3.1. Server Identity

During the TLS negotiation, the client MUST carefully examine the certificate presented by the server to determine if it meets the client's expectations. Particularly, the client MUST check its understanding of the server hostname against the server's identity as presented in the server Certificate message, in order to prevent man-in-the-middle attacks.

Matching is performed according to the rules below (following the example of [RFC4642]):

- o The client MUST use the server hostname it used to open the connection (or the hostname specified in the TLS "server_name" extension [RFC5246]) as the value to compare against the server name as expressed in the server certificate. The client MUST NOT use any form of the server hostname derived from an insecure remote source (e.g., insecure DNS lookup). CNAME canonicalization is not done.

- o If a subjectAltName extension of type dNSName is present in the certificate, it MUST be used as the source of the server's identity.

- o Matching is case-insensitive.

- o A "*" wildcard character MAY be used as the leftmost name component in the certificate. For example, *.example.com would match a.example.com, foo.example.com, etc., but would not match example.com.

- o If the certificate contains multiple names (e.g., more than one dNSName field), then a match with any one of the fields is considered acceptable.

If the match fails, the client MUST either ask for explicit user confirmation or terminate the connection and indicate the server's identity is suspect.

Additionally, clients MUST verify the binding between the identity of the servers to which they connect and the public keys presented by those servers. Clients SHOULD implement the algorithm in Section 6 of [RFC5280] for general certificate validation, but MAY supplement that algorithm with other validation methods that achieve equivalent levels of verification (such as comparing the server certificate against a local store of already-verified certificates and identity

bindings).

If the client has external information as to the expected identity of the server, the hostname check MAY be omitted.

3.2. Client Identity

The server MUST verify the identity of the client to ensure that the incoming client request is legitimate before any configuration or state data is sent to or received from the client.

The NETCONF protocol [RFC6241] requires that the transport protocol's authentication process MUST result in an authenticated client identity whose permissions are known to the server. The authenticated identity of a client is commonly referred to as the NETCONF username.

The username provided by the TLS implementation will be made available to the NETCONF message layer as the NETCONF username without modification. If the username does not comply to the NETCONF requirements on usernames [RFC6241], i.e., the username is not representable in XML, the TLS session MUST be dropped.

Algorithms for mapping certificates or PSK identities (sent by the client) to NETCONF usernames are described below.

3.2.1. Deriving NETCONF Usernames From NETCONF Client Certificates

The algorithm for deriving NETCONF usernames from TLS certificates is patterned after the algorithm for deriving tmSecurityNames from TLS certificates specified in Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP) [RFC6353]. [RFC6353] specifies that an SNMP engine MUST implement several algorithms for transforming a certificate to a tmSecurityName, and lets the SNMP engine deployer choose and configure the algorithm most suitable for the deployer's environment.

When a NETCONF server accepts a TLS connection from a NETCONF client, the NETCONF server MUST produce a NETCONF username from the certificate presented by the NETCONF client. The NETCONF server MAY use any of the following algorithms to produce the NETCONF username from the certificate presented by the NETCONF client:

- o Map a certificate directly to a NETCONF username;

- o Extract the subjectAltName's rfc822Name from the certificate, then use the extracted rfc822Name as the NETCONF username;

- o Extract the subjectAltName’s dnsName from the certificate, then use the extracted dnsName as the NETCONF username;

- o Extract the subjectAltName’s iPAddress from the certificate, then use the extracted iPAddress as the NETCONF username;

- o Examine the subjectAltName’s rfc822Name, dnsName, and iPAddress fields in a pre-defined order, then use the first matching subjectAltName value.

The NETCONF server MUST implement all of these algorithms, and allow the deployer to choose and configure the algorithm used. The certificate-to-username-transforms container in the ietf-netconf-tls-username YANG module specifies how a NETCONF server transforms an certificate into a NETCONF username.

3.2.1.1. Identifying a Certificate

A client certificate has an identity: the certificate. The TLS and corresponding protocols provide an identity. The identity shows that "this client certificate has shown that it, indeed, is on the other side of the connection". With a complete certificate, the certificate receiver can be certain that for someone or something on the other side to use that certificate successfully, it has the associated private key.

The problem with using the entire certificates as the identity is that they are difficult for people to use. It is generally accepted that a fingerprint of a certificate is not likely to come up with a collision against a fingerprint of another (different) certificate. Thus, assuming a good hash algorithm, using fingerprints is a safe way to compare two certificates.

If if a locally held copy of a trusted CA certificate is configured in the transformation container, and that CA certificate was used to validate the path to the presented certificate, then the NETCONF server SHOULD use that list entry in the transformation container. All presented certificates validated by the configured CA certificate will be transformed to NETCONF usernames using the same transformation algorithm.

3.2.1.2. Remote Configuration

The ietf-netconf-tls-username YANG module defines objects for remotely configuring the mapping of TLS certficates to NETCONF usernames.

```
module ietf-netconf-tls-username {

 namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-tls-username";

 prefix "tls-username";

 import ietf-yang-types {
   prefix yang;
 }

 organization
   "IETF NETCONF (Network Configuration) Working Group";

 contact
   "WG Web:   <http://tools.ietf.org/wg/netconf/>
    WG List:  <mailto:netconf@ietf.org>

    WG Chair: Mehmet Ersue
              <mailto:mehmet.ersue@nsn.com>

    WG Chair: Bert Wijnen
              <mailto:bertietf@bwijnen.net>

    Editor:   Mohamad Badra
              <mailto:mbadra@gmail.com>";

 description
   "This module applies to NETCONF over TLS.  It specifies how
    NETCONF servers transform X.509 certificates presented by
    clients into NETCONF usernames.  It also specifies how NETCONF
    clients transform NETCONF usernames into X.509 certificates
    for presentation to NETCONF servers.

    This YANG module is patterned after, and closely models, parts of
    the SNMP-TLS-TM-MIB defined in RFC 6353.  Much of the description
    text has been copied directly from the SNMP-TLS-TM-MIB, and modified
    as necessary.

    Copyright (c) 2011 IETF Trust and the persons identified as
    authors of the code. All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject
    to the license terms contained in, the Simplified BSD
    License set forth in Section 4.c of the IETF Trust’s
    Legal Provisions Relating to IETF Documents
    (http://trustee.ietf.org/license-info).
```

```
    This version of this YANG module is part of RFC XXXX; see
    the RFC itself for full legal notices.";
 // RFC Ed.: replace XXXX with actual RFC number and
 // remove this note

 // RFC Ed.: please update the date to the date of publication

 revision "2011-10-17" {
   description
     "Initial version";
   reference
     "RFC XXXX: NETCONF over Transport Layer Security (TLS)";
 }

 typedef tls-fingerprint-type {
   type binary {
     length "0..255";
   }
   description
     "A fingerprint value that can be used to uniquely reference
      other data of potentially arbitrary length.

      An tls-fingerprint-type value is composed of a 1-octet hashing
      algorithm identifier followed by the fingerprint value.  The
      octet value encoded is taken from the IANA TLS HashAlgorithm
      Registry (RFC 5246).  The remaining octets are filled using the
      results of the hashing algorithm.

      This typedef allows for a zero-length (blank) tls-fingerprint-type
      value for use in containers where the fingerprint value MAY be
      optional.  YANG definitions or implementations MAY refuse to
      accept a zero-length value as appropriate.";
 }

 //
 //  Objects related to deriving NETCONF usernames from  X.509 certificates.
 //
 leaf certificate-to-username-transform-count {
   type yang:gauge32;
   description
     "A count of the number of certificate-to-username-transforms.";
   config false;
 }

 leaf certificate-to-username-transform-last-changed {
   type yang:date-and-time;
   description
     "The date and time when the certificate-to-username-transforms
```

```
      was last modified through any means.  The value 0 means the
      certificate-to-username-transforms has not been modified since
      the NETCONF server was started.";
  config false;
 }

 container certificate-to-username-transforms {
   config true;
   description
     "This container is used by a NETCONF server to map the NETCONF
      client's presented X.509 certificate to a NETCONF username.

      On an incoming TLS connection, the client's presented
      certificate MUST either be validated based on an established
      trust anchor, or it MUST directly match a fingerprint in this
      container.  This container does not provide any mechanisms for
      configuring the trust anchors; the transfer of any needed
      trusted certificates for path validation is expected to occur
      through an out-of-band transfer.

      Once the certificate has been found acceptable (either by path
      validation or directly matching a fingerprint in this container),
      this container is consulted to determine the appropriate
      NETCONF username to identify with the remote connection.  This
      is done by considering each active list entry from this container
      in prioritized order according to its index value.
      Each list entry's certificate-fingerprint value determines
      whether the list entry is a match for the incoming connection:

          1) If the list entry's certificate-fingerprint value
             identifies the presented certificate, then consider the
             list entry as a successful match.

          2) If the list entry's certificate-fingerprint value
             identifies a locally held copy of a trusted CA
             certificate and that CA certificate was used to
             validate the path to the presented certificate, then
             consider the list entry as a successful match.

      Once a matching list entry has been found, the NETCONF server uses
      the map-type value to determine how the NETCONF username associated
      with the session SHOULD be determined.  See the map-type column's
      description for details on determining the NETCONF username
      value.  If it is impossible to determine a NETCONF username from
      the list entry's data combined with the data presented in the
      certificate, then additional list entries MUST be searched looking
      for another potential match.  If a resulting NETCONF username mapped
      from a given list entry is not compatible with the needed requirements
```

```
      of a NETCONF username, then it MUST be considered an invalid match
      and additional list entries MUST be searched looking for another
      potential match.

      If no matching and valid list entry can be found, the connection MUST
      be closed and NETCONF messages MUST NOT be accepted over it.

      Missing values of index are acceptable and implementations
      SHOULD continue to the next highest numbered list entry.  It is
      recommended that administrators skip index values to leave room
      for the insertion of future list entries (for example, use values
      of 10 and 20 when creating initial list entries).

      Users are encouraged to make use of certificates with
      subjectAltName fields that can be used as NETCONF usernames so
      that a single root CA certificate can allow all child certificate's
      subjectAltName to map directly to a NETCONF usernames via a 1:1
      transformation.  However, this container is flexible to allow for
      situations where existing deployed certificate infrastructures do
      not provide adequate subjectAltName values for use as NETCONF
      usernames.";

//        Certificates MAY also be mapped to NETCONF usernames using the
//        CommonName portion of the Subject field.  However, the usage
//        of the CommonName field is deprecated and thus this usage is
//        NOT RECOMMENDED.  Direct mapping from each individual
//        certificate fingerprint to a NETCONF username is also possible
//        but requires one entry in the container per NETCONF username and
//        requires more management operations to completely configure a
//        device.";

   list certificate-to-username-transform {
     key "index";
     description
       "A single list entry that specifies a mapping for an incoming
       TLS certificate to a NETCONF username.";

     leaf index {
       type uint32 {
         range "1..4294967295";
       }
       description
         "A unique, prioritized index for the given entry.  Lower
         numbers indicate a higher priority.";
     }

     leaf certificate-fingerprint {
       type tls-fingerprint-type {
```

```
        length "1..255";
      }
      description
        "A cryptographic hash of a X.509 certificate.  The results of
         a successful matching fingerprint to either the trusted CA in
         the certificate validation path or to the certificate itself
         is dictated by the map-type column.";
    }

    leaf map-type {
      type enumeration {
        enum specified                     { value 1;  }
        enum rfc822Name                    { value 2;  }
        enum dnsName                       { value 3;  }
        enum ipAddress                     { value 4;  }
        enum rfc822Name-dnsName-ipAddress  { value 5;  }
        enum rfc822Name-ipAddress-dnsName  { value 6;  }
        enum dnsName-ipAddress-rfc822Name  { value 7;  }
        enum dnsName-rfc822Name-ipAddress  { value 8;  }
        enum ipAddress-dnsName-rfc822Name  { value 9;  }
        enum ipAddress-rfc822Name-dnsName  { value 10; }
      }

      description
        "Specifies the algorithm for deriving a NETCONF username from
        a certificate.  If a mapping succeeds, then it will return a
        NETCONF username.

        If the resulting mapped value is not compatible with the needed
        requirements of a NETCONF username, then future list entries MUST
        be searched for additional NETCONF username matches to look for a
        mapping that succeeds.

        For each enumerated value listed above, the NETCONF server
        derives the NETCONF from the presented client certificate
        as described:

        specified

          Directly specifies the NETCONF username to be used for
          this certificate.  The value of the NETCONF username
          to use is specified in the data leaf of the list.  The
          data leaf MUST contain a non-zero length value or the
          mapping described in this list entry MUST be considered a
          failure.

        rfc822Name
```

Maps a subjectAltName's rfc822Name to a NETCONF username. The local part of the rfc822Name is passed unaltered but the host-part of the name MUST be passed in lowercase. This mapping results in a 1:1 correspondence between equivalent subjectAltName rfc822Name values and NETCONF username values except that the host-part of the name MUST be passed in lowercase.

Example rfc822Name Field: FooBar@Example.COM
is mapped to NETCONF username: FooBar@example.com.

dnsName

Maps a subjectAltName's dNSName to a NETCONF username after first converting it to all lowercase (RFC 5280 does not specify converting to lowercase so this involves an extra step). This mapping results in a 1:1 correspondence between subjectAltName dNSName values and the NETCONF username values.

reference: RFC 5280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.

ipAddress

Maps a subjectAltName's iPAddress to a NETCONF username by transforming the binary encoded address as follows:

1) for IPv4, the value is converted into a decimal-dotted quad address (e.g., '192.0.2.1').

2) for IPv6 addresses, the value is converted into a 32-character all lowercase hexadecimal string without any colon separators.

   This mapping results in a 1:1 correspondence between subjectAltName iPAddress values and the NETCONF username values.

rfc822Name-dnsName-ipAddress
rfc822Name-ipAddress-dnsName
dnsName-ipAddress-rfc822Name
dnsName-rfc822Name-ipAddress
ipAddress-dnsName-rfc822Name
ipAddress-rfc822Name-dnsName

For each of these enumerations, the NETCONF server derives the NETCONF username in a similar manner. The NETCONF server

```
           derives the NETCONF username from the subjectAltName fields
           rfc822Name, dnsName, and ipAddress, as described in sections
           above.  However, each of these subjectAltName fields is
           examined in the order specified by the enumeration name.
           The first matching subjectAltName value found in the certificate
           MUST be used when deriving the NETCONF username.

           For example, the rfc822Name-dnsName-ipAddress enumeration
           specifies the NETCONF server first examines the rfc822Name,
           then examines the dnsName, then finally examines the ipAddress.
           In contrast, the ipAddress-rfc822Name-dnsName enumeration
           specifies the NETCONF server first examines the ipAddress
           name, then examines the rfc822Name, then finally examines
           the dnsName.

           These mappings result in a 1:1 correspondence between
           subjectAltName values and NETCONF username values.  The
           sub-mapping algorithms produced by these combined algorithms
           cannot produce conflicting results between themselves.";
     }  //  leaf map-type

     leaf data {
       type string {
         length "1..max";
       }
       description
         "Auxiliary data used as optional configuration information for
         a given mapping specified by the map-type column.  Only some
         mapping systems will make use of this column.  The value in this
         column MUST be ignored for any mapping type that does not require
         data present in this column.";
     }
   }  // list certificate-to-username-transform
 }    // container certificate-to-username-transform
}
```

3.2.2. Deriving NETCONF Usernames From PSK identities

Implementations MAY optionally support TLS Pre-Shared Key (PSK) authentication [RFC4279]. RFC4279 describes pre-shared key ciphersuites for TLS. During the TLS Handshake, the client indicates which key to use by including a "PSK identity" in the TLS ClientKeyExchange message [RFC4279]. On the server side, this PSK identity is used to lookup the key corresponding to the presented PSK identity. If the selected pre-shared keys match, then the client is authenticated and the PSK identity is used as the NETCONF username. For details on how the PSK identity MAY be encoded in UTF-8, see

section 5.1. of RFC [RFC6241].

4. Security Considerations

The security considerations described throughout [RFC5246] and [RFC6241] apply here as well.

This document in its current version does not support third-party authentication (e.g., backend Authentication, Authorization, and Accounting (AAA) servers) due to the fact that TLS does not specify this way of authentication and that NETCONF depends on the transport protocol for the authentication service. If third-party authentication is needed, BEEP or SSH transport can be used.

An attacker might be able to inject arbitrary NETCONF messages via some application that does not carefully check exchanged messages. When the :base:1.1 capability is not advertised by both peers, an attacker might be able to deliberately insert the delimiter sequence]]>]]> in a NETCONF message to create a DoS attack. If the :base:1.1 capability is not advertised by both peers, applications and NETCONF APIs MUST ensure that the delimiter sequence]]>]]> never appears in NETCONF messages; otherwise, those messages can be dropped, garbled, or misinterpreted. More specifically, if the delimiter sequence is found in a NETCONF message by the sender side, a robust implementation of this document SHOULD warn the user that illegal characters have been discovered. If the delimiter sequence is found in a NETCONF message by the receiver side (including any XML attribute values, XML comments, or processing instructions), a robust implementation of this document MUST silently discard the message without further processing and then stop the NETCONF session.

Finally, this document does not introduce any new security considerations compared to [RFC6242] and [RFC4742].

5. IANA Considerations

Based on the previous version of this document, RFC 5539, IANA has assigned a TCP port number (6513) in the "Registered Port Numbers" range with the name "netconf-tls". This port will be the default port for NETCONF over TLS, as defined in this document.

Registration Contact: Mohamad Badra, badra@isima.fr.
Transport Protocol: TCP.
Port Number: 6513
Broadcast, Multicast or Anycast: No.
Port Name: netconf-tls.
Service Name: netconf.
Reference: RFC 5539

6. Acknowledgements

A significant amount of the text in Section 3 was lifted from [RFC4642].

The author would like to acknowledge David Harrington, Miao Fuyou, Eric Rescorla, Juergen Schoenwaelder, Simon Josefsson, Olivier Coupelon, Alfred Hoenes, and the NETCONF mailing list members for their comments on the document. The author also appreciates Bert Wijnen, Mehmet Ersue, and Dan Romascanu for their efforts on issues resolving discussion; and Charlie Kaufman, Pasi Eronen, and Tim Polk for the thorough review of this document.

7. Contributor’s Address

Ibrahim Hajjeh
Ineovation
France

EMail: ibrahim.hajjeh@ineovation.fr

Alan Luchuk
SNMP Research, Inc.
3001 Kimberlin Heights Road
Knoxville, TN 37920-9716

EMail: luchuk@snmp.com

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4279] Eronen, P. and H. Tschofenig, "Pre-Shared Key Ciphersuites

for Transport Layer Security (TLS)", RFC 4279, December 2005.

[RFC4742] Wasserman, M. and T. Goddard, "Using the NETCONF Configuration Protocol over Secure SHell (SSH)", RFC 4742, December 2006.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC5539] Badra, M., "NETCONF over Transport Layer Security (TLS)", RFC 5539, May 2009.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.

[RFC6353] Hardaker, W., "Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP)", RFC 6353, July 2011.

8.2. Informative References

[RFC4642] Murchison, K., Vinocur, J., and C. Newman, "Using Transport Layer Security (TLS) with Network News Transfer Protocol (NNTP)", RFC 4642, October 2006.

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, July 2008.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

Appendix A. Change Log (to be removed by RFC Editor before publication)

A.1. From RFC5539 to draft-badra-netconf-rfc5539bis-00

- o Added text on how the generation of a NETCONF username is done.
- o Added text on how does this document fulfill the requirements in 6241 for the format of the username.

- o Removed unneeded wording about client/server, and changed use of client/server, manager/agent to SSH client/server and NETCONF client/server.
- o Added text to Security Considerations about EOM issues.
- o Added option for the chunked encoding described in RFC6242.
- o Added 1.1 capability to enable the chunked encoding described in RFC6242.

Author’s Address

Mohamad Badra
DU

Email: mbadra@gmail.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: June 25, 2012

A. Bierman
Brocade
M. Bjorklund
Tail-f Systems

December 23, 2011

Network Configuration Protocol (NETCONF) Access Control Model
draft-ietf-netconf-access-control-07

Abstract

The standardization of network configuration interfaces for use with the NETCONF protocol requires a structured and secure operating environment that promotes human usability and multi-vendor interoperability. There is a need for standard mechanisms to restrict NETCONF protocol access for particular users to a pre-configured subset of all available NETCONF protocol operations and content. This document defines such an access control model.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on June 25, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 4
1.1. Terminology 4
2. Access Control Design Objectives 6
2.1. Access Control Points 6
2.2. Simplicity 7
2.3. Procedural Interface 7
2.4. Datastore Access 7
2.5. Users and Groups 7
2.6. Maintenance 8
2.7. Configuration Capabilities 8
2.8. Identifying Security-Sensitive Content 8
3. NETCONF Access Control Model (NACM) 10
3.1. Introduction 10
3.1.1. Features 10
3.1.2. External Dependencies 11
3.1.3. Message Processing Model 11
3.2. Datastore Access 13
3.2.1. Access Rights 13
3.2.2. <get> and <get-config> Operations 14
3.2.3. <edit-config> Operation 14
3.2.4. <copy-config> Operation 15
3.2.5. <delete-config> Operation 16
3.2.6. <commit> Operation 16
3.2.7. <discard-changes> Operation 16
3.2.8. <kill-session> Operation 16
3.3. Model Components 16
3.3.1. Users 17
3.3.2. Groups 17
3.3.3. Emergency Recovery Session 17
3.3.4. Global Enforcement Controls 17
3.3.4.1. enable-nacm Switch 17
3.3.4.2. read-default Switch 17
3.3.4.3. write-default Switch 18
3.3.4.4. exec-default Switch 18
3.3.4.5. enable-external-groups Switch 18
3.3.5. Access Control Rules 19
3.4. Access Control Enforcement Procedures 19
3.4.1. Initial Operation 19
3.4.2. Session Establishment 20
3.4.3. "access-denied" Error Handling 20
3.4.4. Incoming RPC Message Validation 20

3.4.5. Data Node Access Validation 23
3.4.6. Outgoing <notification> Authorization 25
3.5. Data Model Definitions 27
3.5.1. Data Organization 27
3.5.2. YANG Module 28
3.6. IANA Considerations 38
3.7. Security Considerations 39
3.7.1. NACM Configuration and Monitoring Considerations . . . 39
3.7.2. General Configuration Issues 40
3.7.3. Data Model Design Considerations 42
4. References 43
4.1. Normative References 43
4.2. Informative References 43
Appendix A. Usage Examples 44
A.1. <groups> Example 44
A.2. Module Rule Example 45
A.3. RPC Rule Example 46
A.4. Data Rule Example 48
A.5. Notification Rule Example 50
Appendix B. Change Log 52
B.1. 06-07 52
B.2. 05-06 52
B.3. 04-05 52
B.4. 03-04 52
B.5. 02-03 53
B.6. 01-02 53
B.7. 00-01 53
B.8. 00 53
Authors' Addresses 54

1. Introduction

The NETCONF protocol does not provide any standard mechanisms to restrict the protocol operations and content that each user is authorized to access.

There is a need for inter-operable management of the controlled access to administrator selected portions of the available NETCONF content within a particular server.

This document addresses access control mechanisms for the Operation and Content layers of NETCONF, as defined in [RFC6241]. It contains three main sections:

1. Access Control Design Objectives

2. NETCONF Access Control Model (NACM)

3. YANG Data Model (ietf-netconf-acm.yang)

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The following terms are defined in [RFC6241] and are not redefined here:

- client

- datastore

- protocol operation

- server

- session

- user

The following terms are defined in [RFC6020] and are not redefined here:

- data node

- data definition statement

The following terms are used throughout this documentation:

access control: A security feature provided by the NETCONF server, that allows an administrator to restrict access to a subset of all NETCONF protocol operations and data, based on various criteria.

access control model (ACM): A conceptual model used to configure and monitor the access control procedures desired by the administrator to enforce a particular access control policy.

access control rule: The criteria used to determine if a particular NETCONF protocol operation will be permitted or denied.

access operation: How a request attempts to access a conceptual object. One of "none", "read", "create", "delete", "update", and "execute".

recovery session: A special administrative session that is given unlimited NETCONF access, and is exempt from all access control enforcement. The mechanism(s) used by a server to control and identify whether a session is a recovery session or not are implementation-specific and outside the scope of this document.

write access: A shorthand for the "create", "delete", and "update" access operations.

2. Access Control Design Objectives

This section documents the design objectives for the NETCONF Access Control Model presented in Section 3.

2.1. Access Control Points

NETCONF allows new protocol operations to be added at any time, and the YANG data modeling language supports this feature. It is not possible to design an ACM for NETCONF that only focuses on a static set of protocol operations, like some other protocols. Since few assumptions can be made about an arbitrary protocol operation, the NETCONF architectural server components need to be protected at three conceptual control points.

```
                +-------------+                 +-------------+
 client         |  protocol   |                 |  data node  |
 request -->    |  operation  | ------------->  |   access    |
                |  allowed?   |   datastore     |  allowed?   |
                +-------------+   or state      +-------------+
                                  data access


                +----------------+
                |  notification  |
 event -->      |  allowed?      |
                +----------------+
```

Figure 1

The following access control points, described in Figure 1, are identified:

protocol operation: Permission to invoke specific protocol operations.

datastore: Permission to read and/or alter specific data nodes within any datastore.

notification: Permission to receive specific notification event types.

2.2. Simplicity

There is concern that a complicated ACM will not be widely deployed, because it is too hard to use. It needs to be easy to do simple things, and possible to do complex things, instead of hard to do everything.

Configuration of the access control system needs to be as simple as possible. Simple and common tasks need to be easy to configure, and require little expertise or domain-specific knowledge. Complex tasks are possible using additional mechanisms, which may require additional expertise.

A single set of access control rules ought to be able to control all types of NETCONF protocol operation invocation, all datastore access, and all notification events.

Access control ought to be defined with a small and familiar set of permissions, while still allowing full control of NETCONF datastore access.

2.3. Procedural Interface

The NETCONF protocol uses a remote procedure call model, and an extensible set of protocol operations. Access control for any possible protocol operation is necessary.

2.4. Datastore Access

It is necessary to control access to specific nodes and subtrees within the NETCONF datastore, regardless of which protocol operation, standard or proprietary, was used to access the datastore.

2.5. Users and Groups

It is necessary that access control rules for a single user or a configurable group of users can be configured.

The ACM needs to support the concept of administrative groups, to support the well-established distinction between a root account and other types of less-privileged conceptual user accounts. These groups need to be configurable by the administrator.

It is necessary that the user-to-group mapping can be delegated to a central server, such as a RADIUS server [RFC2865] [RFC5607]. Since authentication is performed by the NETCONF transport layer, and RADIUS performs authentication and service authorization at the same time, the underlying NETCONF transport needs to be able to report a

set of group names associated with the user to the server. It is necessary that the administrator can disable the usage of these group names within the ACM.

2.6. Maintenance

It ought to be possible to disable part or all of the access control model enforcement procedures without deleting any access control rules.

2.7. Configuration Capabilities

Suitable configuration and monitoring mechanisms are needed to allow an administrator to easily manage all aspects of the ACM behavior. A standard data model, suitable for use with the <edit-config> protocol operation needs to be available for this purpose.

Access control rules to restrict access operations on specific subtrees within the configuration datastore need to be supported.

2.8. Identifying Security-Sensitive Content

One of the most important aspects of the data model documentation, and biggest concerns during deployment, is the identification of security-sensitive content. This applies to protocol operations in NETCONF, not just data and notifications.

It is mandatory for security-sensitive objects to be documented in the Security Considerations section of an RFC. This is nice, but it is not good enough, for the following reasons:

- This documentation-only approach forces administrators to study the RFC and determine if there are any potential security risks introduced by a new data model.

- If any security risks are identified, then the administrator can study some more RFC text, and determine how to mitigate the security risk(s).

- The ACM on each server can be configured to mitigate the security risks, e.g., require privileged access to read or write the specific data identified in the Security Considerations section.

- If the ACM is not pre-configured, then there will be a time window of vulnerability, after the new data model is loaded, and before the new access control rules for that data model are configured, enabled, and debugged.

Often, the administrator just wants to disable default access to the secure content, so no inadvertent or malicious changes can be made to the server. This allows the default rules to be more lenient, without significantly increasing the security risk.

A data model designer needs to be able to use machine-readable statements to identify NETCONF content which needs to be protected by default. This will allow client and server tools to automatically identify data-model specific security risks, by denying access to sensitive data unless the user is explicitly authorized to perform the requested access operation.

3. NETCONF Access Control Model (NACM)

3.1. Introduction

This section provides a high-level overview of the access control model structure. It describes the NETCONF protocol message processing model, and the conceptual access control requirements within that model.

3.1.1. Features

The NACM data model provides the following features:

- o Independent control of RPC, data, and notification access.
- o Simple access control rules configuration data model that is easy to use.
- o The concept of an emergency recovery session is supported, but configuration of the server for this purpose is beyond the scope of this document. An emergency recovery session will bypass all access control enforcement, in order to allow it to initialize or repair the NACM configuration.
- o A simple and familiar set of datastore permissions is used.
- o Support for YANG security tagging (e.g., "nacm:default-deny-write" statement) allows default security modes to automatically exclude sensitive data.
- o Separate default access modes for read, write, and execute permissions.
- o Access control rules are applied to configurable groups of users.
- o The access control enforcement procedures can be disabled during operation, without deleting any access control rules, in order to debug operational problems.
- o Access control rules are simple to configure.
- o The number of denied protocol operation requests and denied datastore write requests can be monitored by the client.
- o Simple unconstrained YANG instance identifiers are used to configure access control rules for specific data nodes.

3.1.2. External Dependencies

The NETCONF [RFC6241] protocol is used for all management purposes within this document.

The YANG Data Modeling Language [RFC6020] is used to define the NETCONF data models specified in this document.

3.1.3. Message Processing Model

The following diagram shows the conceptual message flow model, including the points at which access control is applied, during NETCONF message processing.

```
                  +-------------------------+
                  |       session           |
                  |      (username)         |
                  +-------------------------+
                     |                 ^
                     V                 |
              +--------------+     +---------------+
              |   message    |     |   message     |
              | dispatcher   |     |   generator   |
              +--------------+     +---------------+
                   |                  ^         ^
                   V                  |         |
           +===========+     +-------------+   +----------------+
           |   <rpc>   |---> | <rpc-reply> |   | <notification> |
           | acc. ctl  |     |  generator  |   |  generator     |
           +===========+     +-------------+   +----------------+
                 |              ^    ^                ^
                 V       +------+    |                |
           +-----------+ |   +=============+  +================+
           |   <rpc>   | |   |    read     |  | <notification> |
           | processor |-+   | data node   |  |  access ctl    |
           |           |     | acc. ctl    |  |                |
           +-----------+     +=============+  +================+
                 |   |                  ^        ^
                 V   +----------------+ |        |
           +===========+              | |        |
           |  write    |              | |        |
           | data node |              | |        |
           | acc. ctl  | -----------+ | |        |
           +===========+            | | |        |
                 |                  | | |        |
                 V                  V V |        |
           +---------------+      +-----------------+
           | configuration | ---> |     server      |
           |   datastore   |      | instrumentation |
           |               | <--- |                 |
           +---------------+      +-----------------+
```

Figure 2

The following high-level sequence of conceptual processing steps is executed for each received <rpc> message, if access control enforcement is enabled:

o Access control is applied to all <rpc> messages (except <close-session>) received by the server, individually, for each active

session, unless the session is identified as a "recovery session".

- o If the user is authorized to execute the specified protocol operation, then processing continues, otherwise the request is rejected with an "access-denied" error.

- o If the configuration datastore or conceptual state data is accessed by the protocol operation, then the server checks if the client is authorized to access the nodes in the data store. If the user is authorized to perform the requested access operation on the requested data, then processing continues.

The following sequence of conceptual processing steps is executed for each generated notification event, if access control enforcement is enabled:

- o Server instrumentation generates a notification, for a particular subscription.

- o The notification access control enforcer checks the notification event type, and if it is one which the user is not authorized to read, then the notification is dropped for that subscription.

3.2. Datastore Access

The same access control rules apply to all datastores. For example, the candidate configuration datastore or the running configuration datastore.

Only the standard NETCONF datastores (candidate, running, and startup) are controlled by NACM. Local or remote files or datastores accessed via the <url> parameter are not controlled by NACM.

3.2.1. Access Rights

A small set of hard-wired datastore access rights is needed to control access to all possible NETCONF protocol operations, including vendor extensions to the standard protocol operation set.

The "CRUDX" model can support all NETCONF protocol operations:

- o Create: Allows the client to add a new data node instance to a datastore.

- o Read: Allows the client to read a data node instance from a datastore, or receive the notification event type.

- o Update: Allows the client to update an existing data node instance in a datastore.

- o Delete: Allows the client to delete a data node instance from a datastore.

- o eXec: Allows the client to execute the protocol operation.

3.2.2. <get> and <get-config> Operations

Data nodes to which the client does not have read access are silently omitted from the <rpc-reply> message. This is done to allow NETCONF filters for <get> and <get-config> to function properly, instead of causing an "access-denied" error because the filter criteria would otherwise include unauthorized read access to some data nodes. For NETCONF filtering purposes, the selection criteria is applied to the subset of nodes that the user is authorized to read, not the entire datastore.

3.2.3. <edit-config> Operation

The NACM access rights are not directly coupled to the <edit-config> "operation" attribute, although they are similar. Instead, a NACM access right applies to all protocol operations which would result in a particular access operation to the target datastore. This section describes how these access rights apply to the specific access operations supported by the <edit-config> protocol operation.

If the effective access operation is "none" (i.e., default-operation="none") for a particular data node, then no access control is applied to that data node. This is required to allow access to a sub-tree within larger data structure. For example, a user may be authorized to create a new "/interfaces/interface" list entry, but not be authorized to create or delete its parent container ("/interfaces"). If the "/interfaces" container already exists in the target datastore, then the effective operation will be "none" for the "/interfaces" node if an "/interfaces/interface" list entry is edited.

If the protocol operation would result in the creation of a data store node, and the user does not have "create" access permission for that node, the protocol operation is rejected with an "access-denied" error.

If the protocol operation would result in the deletion of a data store node, and the user does not have "delete" access permission for that node, the protocol operation is rejected with an "access-denied" error.

If the protocol operation would result in the modification of a data store node, and the user does not have "update" access permission for that node, the protocol operation is rejected with an "access-denied" error.

A "merge" or "replace" <edit-config> operation may include data nodes which do not alter portions of the existing datastore. For example, a container or list node may be present for naming purposes, but does not actually alter the corresponding datastore node. These unaltered data nodes are ignored by the server, and do not require any access rights by the client.

A "merge" <edit-config> operation may include data nodes, but not include particular child data nodes that are present in the datastore. These missing data nodes within the scope of a "merge" <edit-config> operation are ignored by the server, and do not require any access rights by the client.

The contents of specific restricted datastore nodes MUST NOT be exposed in any <rpc-error> elements within the reply.

3.2.4. <copy-config> Operation

Access control for the <copy-config> protocol operation requires special consideration because the administrator may be replacing the entire target datastore.

If the source of the <copy-config> protocol operation is the running configuration datastore, and the target is the startup configuration datastore, the client is only required to have permission to execute the <copy-config> protocol operation.

Otherwise:

- If the source of the <copy-config> operation is a datastore, then data nodes to which the client does not have read access are silently omitted.

- If the target of the <copy-config> operation is a datastore, the client needs access to the modified nodes. Specifically:

  If the protocol operation would result in the creation of a data store node, and the user does not have "create" access permission for that node, the protocol operation is rejected with an "access-denied" error.

  If the protocol operation would result in the deletion of a data store node, and the user does not have "delete" access

permission for that node, the protocol operation is rejected with an "access-denied" error.

If the protocol operation would result in the modification of a data store node, and the user does not have "update" access permission for that node, the protocol operation is rejected with an "access-denied" error.

3.2.5. <delete-config> Operation

Access to the <delete-config> protocol operation is denied by default. The 'exec-default' parameter does not apply to this protocol operation. Access control rules must be explicitly configured to allow invocation by a non-recovery session.

3.2.6. <commit> Operation

The server MUST determine the exact nodes in the running configuration datastore which are actually different, and only check "create", "update", and "delete" access permissions for this set of nodes, which could be empty.

For example, if a session can read the entire datastore, but only change one leaf, that session needs to be able to edit and commit that one leaf.

3.2.7. <discard-changes> Operation

The client is only required to have permission to execute the <discard-changes> protocol operation. No datastore permissions are needed.

3.2.8. <kill-session> Operation

The <kill-session> operation does not directly alter a datastore. However, it allows one session to disrupt another session which is editing a datastore.

Access to the <kill-session> protocol operation is denied by default. The 'exec-default' parameter does not apply to this protocol operation. Access control rules must be explicitly configured to allow invocation by a non-recovery session.

3.3. Model Components

This section defines the conceptual components related to access control model.

3.3.1. Users

A "user" is the conceptual entity that is associated with the access permissions granted to a particular session. A user is identified by a string which is unique within the server.

As described in [RFC6241], the user name string is derived from the transport layer during session establishment. If the transport layer cannot authenticate the user, the session is terminated.

3.3.2. Groups

Access to a specific NETCONF protocol operation is granted to a session, associated with a group, not a user.

A group is identified by its name. All group names are unique within the server.

A group member is identified by a user name string.

The same user can be a member of multiple groups.

3.3.3. Emergency Recovery Session

The server MAY support a "recovery session" mechanism, which will bypass all access control enforcement. This is useful for restricting initial access and repairing a broken access control configuration.

3.3.4. Global Enforcement Controls

There are five global controls that are used to help control how access control is enforced.

3.3.4.1. enable-nacm Switch

A global "enable-nacm" on/off switch is provided to enable or disable all access control enforcement. When this global switch is set to "true", then all requests are checked against the access control rules, and only permitted if configured to allow the specific access request. When this global switch is set to "false", then all access requested are permitted.

3.3.4.2. read-default Switch

An on/off "read-default" switch is provided to enable or disable default access to receive data in replies and notifications. When the "enable-nacm" global switch is set to "true", then this global

switch is relevant, if no matching access control rule is found to explicitly permit or deny read access to the requested NETCONF datastore data or notification event type.

When this global switch is set to "permit", and no matching access control rule is found for the NETCONF datastore read or notification event requested, then access is permitted.

When this global switch is set to "deny", and no matching access control rule is found for the NETCONF datastore read or notification event requested, then access is denied.

3.3.4.3. write-default Switch

An on/off "write-default" switch is provided to enable or disable default access to alter configuration data. When the "enable-nacm" global switch is set to "true", then this global switch is relevant, if no matching access control rule is found to explicitly permit or deny write access to the requested NETCONF datastore data.

When this global switch is set to "permit", and no matching access control rule is found for the NETCONF datastore write requested, then access is permitted.

When this global switch is set to "deny", and no matching access control rule is found for the NETCONF datastore write requested, then access is denied.

3.3.4.4. exec-default Switch

An on/off "exec-default" switch is provided to enable or disable default access to execute protocol operations. When the "enable-nacm" global switch is set to "true", then this global switch is relevant, if no matching access control rule is found to explicitly permit or deny access to the requested NETCONF protocol operation.

When this global switch is set to "permit", and no matching access control rule is found for the NETCONF protocol operation requested, then access is permitted.

When this global switch is set to "deny", and no matching access control rule is found for the NETCONF protocol operation requested, then access is denied.

3.3.4.5. enable-external-groups Switch

When this global switch is set to "true", the group names reported by the NETCONF transport layer for a session are used together with the

locally configured group names, to determine the access control rules for the session.

When this switch is set to "false", the group names reported by the NETCONF transport layer are ignored by NACM.

3.3.5. Access Control Rules

There are 4 types of rules available in NACM:

module rule: Controls access for definitions in a specific YANG module, identified by its name.

protocol operation rule: Controls access for a specific protocol operation, identified by its YANG module and name.

data node rule: Controls access for a specific data node, identified by its path location within the conceptual XML document for the data node.

notification rule: Controls access for a specific notification event type, identified by its YANG module and name.

3.4. Access Control Enforcement Procedures

There are seven separate phases that need to be addressed, four of which are related to the NETCONF message processing model. In addition, the initial start-up mode for a NETCONF server, session establishment, and "access-denied" error handling procedures also need to be considered.

The server MUST use the access control rules in effect at the time it starts processing the message. The same access control rules MUST stay in effect for the processing of the entire message.

3.4.1. Initial Operation

Upon the very first start-up of the NETCONF server, the access control configuration will probably not be present. If it isn’t, a server MUST NOT allow any write access to any session role except a "recovery session".

Access rules are enforced any time a request is initiated from a user session. Access control is not enforced for server-initiated access requests, such as the initial load of the running datastore, during bootup.

3.4.2. Session Establishment

The access control model applies specifically to the well-formed XML content transferred between a client and a server, after session establishment has been completed, and after the <hello> exchange has been successfully completed.

Once session establishment is completed, and a user has been authenticated, the NETCONF transport layer reports the user name and a possibly empty set of group names associated with the user to the NETCONF server. The NETCONF server will enforce the access control rules, based on the supplied user name, group names, and the configuration data stored on the server.

3.4.3. "access-denied" Error Handling

The "access-denied" error-tag is generated when the access control system denies access to either a request to invoke a protocol operation or a request to perform a particular access operation on the configuration datastore.

A server MUST NOT include any information the client is not allowed to read in any <error-info> elements within the <rpc-error> response.

3.4.4. Incoming RPC Message Validation

The diagram below shows the basic conceptual structure of the access control processing model for incoming NETCONF <rpc> messages, within a server.

```
                   NETCONF server
                  +------------+
                  |    XML     |
                  |   message  |
                  | dispatcher |
                  +------------+
                         |
                         |
                         V
                  +------------+
                  | NC-base NS |
                  |   <rpc>    |
                  +------------+
                    |   |  |
                    |   |  +-------------------------+
                    |   +------------+               |
                    V                V               V
               +-----------+ +---------------+ +------------+
               | Vendor NS | | NC-base NS    | | NC-base NS |
               | <my-edit> | | <edit-config> | | <unlock>   |
               +-----------+ +---------------+ +------------+
                      |               |
                      |               |
                      V               V
                    +----------------------+
                    |                      |
                    |    configuration     |
                    |      datastore       |
                    +----------------------+
```

Figure 3

Access control begins with the message dispatcher.

After the server validates the <rpc> element, and determines the namespace URI and the element name of the protocol operation being requested, the server verifies that the user is authorized to invoke the protocol operation.

The server MUST separately authorize every protocol operation by following these steps:

1. If the "enable-nacm" leaf is set to "false", then the protocol operation is permitted.

2. If the requesting session is identified as a "recovery session", then the protocol operation is permitted.

3. If the requested operation is the NETCONF <close-session> protocol operation, then the protocol operation is permitted.

4. Check all the "group" entries for ones that contain a "user-name" entry that equals the user name for the session making the request. If the "enable-external-groups" leaf is "true", add to these groups the set of groups provided by the transport layer.

5. If no groups are found, continue with step 10.

6. Process all rule-list entries, in the order they appear in the configuration. If a rule-list's "group" leaf-list does not match any of the user's groups, proceed to the next rule-list entry.

7. For each rule-list entry found, process all rules, in order, until a rule that matches the requested access operation is found. A rule matches if all of the following criteria are met:

   * The rule's "module-name" leaf is "*", or equals the name of the YANG module where the protocol operation is defined.

   * The rule does not have a "rule-type" defined, or the "rule-type" is "protocol-operation" and the "rpc-name" is "*" or equals the name of the requested protocol operation.

   * The rule's "access-operations" leaf has the "exec" bit set, or has the special value "*".

8. If a matching rule is found, then the "action" leaf is checked. If it is equal to "permit", then the protocol operation is permitted, otherwise it is denied.

9. Otherwise, no matching rule was found in any rule-list entry.

10. If the requested protocol operation is defined in a YANG module advertised in the server capabilities, and the "rpc" statement contains a "nacm:default-deny-all" statement, then the protocol operation is denied.

11. If the requested protocol operation is the NETCONF <kill-session> or <delete-config>, then the protocol operation is denied.

12. If the "exec-default" leaf is set to "permit", then permit the protocol operation, otherwise deny the request.

If the user is not authorized to invoke the protocol operation then an <rpc-error> is generated with the following information:

error-tag: access-denied

error-path: Identifies the requested protocol operation. For example:

```
<error-path
  xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    /nc:rpc/nc:edit-config
</error-path>
```

represents the <edit-config> protocol operation in the NETCONF base namespace.

If a datastore is accessed, either directly or as a side effect of the protocol operation, then the server MUST intercept the access operation and make sure the user is authorized to perform the requested access operation on the specified data, as defined in Section 3.4.5.

3.4.5. Data Node Access Validation

If a data node within a datastore is accessed, then the server MUST ensure that the user is authorized to perform the requested read, create, update, or delete access operation on the specified data node.

The data node access request is authorized by following these steps:

1. If the "enable-nacm" leaf is set to "false", then the access operation is permitted.

2. If the requesting session is identified as a "recovery session", then the access operation is permitted.

3. Check all the "group" entries for ones that contain a "user-name" entry that equals the user name for the session making the request. If the the "enable-external-groups" leaf is "true", add to these groups the set of groups provided by the transport layer.

4. If no groups are found, continue with step 9.

5. Process all rule-list entries, in the order they appear in the configuration. If a rule-list's "group" leaf-list does not match any of the user's groups, proceed to the next rule-list entry.

6. For each rule-list entry found, process all rules, in order, until a rule that matches the requested access operation is found. A rule matches if all of the following criteria are met:

   * The rule's "module-name" leaf is "*", or equals the name of the YANG module where the requested data node is defined.

   * The rule does not have a "rule-type" defined, or the "rule-type" is "data-node" and the "path" matches the requested data node.

   * For a read access operation, the rule's "access-operations" leaf has the "read" bit set, or has the special value "*".

   * For a create access operation, the rule's "access-operations" leaf has the "create" bit set, or has the special value "*".

   * For a delete access operation, the rule's "access-operations" leaf has the "delete" bit set, or has the special value "*".

   * For an update access operation, the rule's "access-operations" leaf has the "update" bit set, or has the special value "*".

7. If a matching rule is found, then the "action" leaf is checked. If it is equal to "permit", then the data node access is permitted, otherwise it is denied. For a read access operation, "denied" means that the requested data is not returned in the reply.

8. Otherwise, no matching rule was found in any rule-list entry.

9. For a read access operation, if the requested data node is defined in a YANG module advertised in the server capabilities, and the data definition statement contains a "nacm:default-deny-all" statement, then the requested data node is not included in the reply.

10. For a write access operation, if the requested data node is defined in a YANG module advertised in the server capabilities, and the data definition statement contains a "nacm:default-deny-

write" or a "nacm:default-deny-all" statement, then the data node access request is denied.

11. For a read access operation, if the "read-default" leaf is set to "permit", then include the requested data node in the reply, otherwise do not include the requested data node in the reply.

12. For a write access operation, if the "write-default" leaf is set to "permit", then permit the data node access request, otherwise deny the request.

3.4.6. Outgoing <notification> Authorization

Configuration of access control rules specifically for descendant nodes of the notification event type element are outside the scope of this document. If the user is authorized to receive the notification event type, then it is also authorized to receive any data it contains.

The following figure shows the conceptual message processing model for outgoing <notification> messages.

```
                 NETCONF server
                +------------+
                |    XML     |
                |   message  |
                | generator  |
                +------------+
                      ^
                      |
               +----------------+
               | <notification> |
               |  generator     |
               +----------------+
                      ^
                      |
              +=================+
              | <notification>  |
              |  access control |
              |  <eventType>    |
              +=================+
                      ^
                      |
           +------------------------+
           | server instrumentation |
           +------------------------+
                     |     ^
                     V     |
            +----------------------+
            |    configuration     |
            |      datastore       |
            +----------------------+
```

Figure 4

The generation of a notification for a specific subscription [RFC5277] is authorized by following these steps:

1. If the "enable-nacm" leaf is set to "false", then the notification is permitted.

2. If the session is identified as a "recovery session", then the notification is permitted.

3. If the notification is the NETCONF <replayComplete> or <notificationComplete> event type [RFC5277], then the notification is permitted.

4. Check all the "group" entries for ones that contain a "user-name" entry that equals the user name for the session making the request. If the "enable-external-groups" leaf is "true", add to these groups the set of groups provided by the transport layer.

5. If no groups are found, continue with step 10.

6. Process all rule-list entries, in the order they appear in the configuration. If a rule-list's "group" leaf-list does not match any of the user's groups, proceed to the next rule-list entry.

7. For each rule-list entry found, process all rules, in order, until a rule that matches the requested access operation is found. A rule matches if all of the following criteria are met:

   * The rule's "module-name" leaf is "*", or equals the name of the YANG module where the notification is defined.

   * The rule does not have a "rule-type" defined, or the "rule-type" is "notification" and the "notification-name" is "*", equals the name of the notification.

   * The rule's "access-operations" leaf has the "read" bit set, or has the special value "*".

8. If a matching rule is found, then the "action" leaf is checked. If it is equal to "permit", then permit the notification, otherwise drop the notification for the associated subscription.

9. Otherwise, no matching rule was found in any rule-list entry.

10. If the requested notification is defined in a YANG module advertised in the server capabilities, and the "notification" statement contains a "nacm:default-deny-all" statement, then the notification is dropped for the associated subscription.

11. If the "read-default" leaf is set to "permit", then permit the notification, otherwise drop the notification for the associated subscription.

3.5. Data Model Definitions

3.5.1. Data Organization

The following diagram highlights the contents and structure of the NACM YANG module.

```
      +--rw nacm
         +--rw enable-nacm?            boolean
         +--rw read-default?           action-type
         +--rw write-default?          action-type
         +--rw exec-default?           action-type
         +--rw enable-external-groups? boolean
         +--ro denied-operations       yang:zero-based-counter32
         +--ro denied-data-writes      yang:zero-based-counter32
         +--ro denied-notifications    yang:zero-based-counter32
         +--rw groups
         |  +--rw group [name]
         |     +--rw name         group-name-type
         |     +--rw user-name*   user-name-type
         +--rw rule-list [name]
            +--rw name     string
            +--rw group*   union
            +--rw rule [name]
               +--rw name                 string
               +--rw module-name?         union
               +--rw (rule-type)?
               |  +--:(protocol-operation)
               |  |  +--rw rpc-name?            union
               |  +--:(notification)
               |  |  +--rw notification-name?   union
               |  +--:(data-node)
               |     +--rw path                 node-instance-identifier
               +--rw access-operations?   union
               +--rw action               action-type
               +--rw comment?             string
```

3.5.2. YANG Module

The following YANG module specifies the normative NETCONF content that MUST by supported by the server.

The "ietf-netconf-acm" YANG module imports typedefs from [RFC6021].

```
// RFC Ed.: please update the date to the date of publication
<CODE BEGINS> file="ietf-netconf-acm@2011-12-23.yang"

module ietf-netconf-acm {

  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-acm";

  prefix "nacm";

  import ietf-yang-types {
```

```
      prefix yang;
    }

    organization
      "IETF NETCONF (Network Configuration) Working Group";


    contact
      "WG Web:   <http://tools.ietf.org/wg/netconf/>
       WG List:  <mailto:netconf@ietf.org>

       WG Chair: Mehmet Ersue
                 <mailto:mehmet.ersue@nsn.com>

       WG Chair: Bert Wijnen
                 <mailto:bertietf@bwijnen.net>

       Editor:   Andy Bierman
                 <mailto:andy@netconfcentral.org>

       Editor:   Martin Bjorklund
                 <mailto:mbj@tail-f.com>";

    description
      "NETCONF Access Control Model.

       Copyright (c) 2011 IETF Trust and the persons identified as
       authors of the code. All rights reserved.

       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject
       to the license terms contained in, the Simplified BSD
       License set forth in Section 4.c of the IETF Trust's
       Legal Provisions Relating to IETF Documents
       (http://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC XXXX; see
       the RFC itself for full legal notices.";
    // RFC Ed.: replace XXXX with actual RFC number and
    // remove this note

    // RFC Ed.: remove this note
    // Note: extracted from draft-ietf-netconf-access-control-07.txt

    // RFC Ed.: please update the date to the date of publication
    revision "2011-12-23" {
      description
        "Initial version";
```

```
      reference
        "RFC XXXX: Network Configuration Protocol
                   Access Control Model";
    }

    /*
     * Extension statements
     */

    extension default-deny-write {
      description
        "Used to indicate that the data model node
         represents a sensitive security system parameter.

         If present, and the NACM module is enabled (i.e.,
         /nacm/enable-nacm object equals 'true'), the NETCONF server
         will only allow the designated 'recovery session' to have
         write access to the node.  An explicit access control rule is
         required for all other users.

         The 'default-deny-write' extension MAY appear within a data
         definition statement.  It is ignored otherwise.";
    }

    extension default-deny-all {
      description
        "Used to indicate that the data model node
         controls a very sensitive security system parameter.

         If present, and the NACM module is enabled (i.e.,
         /nacm/enable-nacm object equals 'true'), the NETCONF server
         will only allow the designated 'recovery session' to have
         read, write, or execute access to the node.  An explicit
         access control rule is required for all other users.

         The 'default-deny-all' extension MAY appear within a data
         definition statement, 'rpc' statement, or 'notification'
         statement.  It is ignored otherwise.";
    }

    /*
     * Derived types
     */

    typedef user-name-type {
      type string {
        length "1..max";
      }
```

```
      description
        "General Purpose User Name string.";
    }

    typedef matchall-string-type {
      type string {
        pattern "\*";
      }
      description
        "The string containing a single asterisk '*' is used
         to conceptually represent all possible values
         for the particular leaf using this data type.";
    }

    typedef access-operations-type {
      type bits {
        bit create {
          description
            "Any protocol operation that creates a
             new data node.";
        }
        bit read {
          description
            "Any protocol operation or notification that
             returns the value of a data node.";
        }
        bit update {
          description
            "Any protocol operation that alters an existing
             data node.";
        }
        bit delete {
          description
            "Any protocol operation that removes a data node.";
        }
        bit exec {
          description
            "Execution access to the specified protocol operation.";
        }
      }
      description
        "NETCONF Access Operation.";
    }

    typedef group-name-type {
      type string {
        length "1..max";
        pattern "[^\*].*";
```

```
      }
      description
        "Name of administrative group to which
         users can be assigned.";
    }

    typedef action-type {
      type enumeration {
        enum permit {
          description
            "Requested action is permitted.";
        }
        enum deny {
          description
            "Requested action is denied.";
        }
      }
      description
        "Action taken by the server when a particular
         rule matches.";
    }

    typedef node-instance-identifier {
      type yang:xpath1.0;
      description
        "Path expression used to represent a special
         data node instance identifier string.

         A node-instance-identifier value is an
         unrestricted YANG instance-identifier expression.
         All the same rules as an instance-identifier apply
         except predicates for keys are optional.  If a key
         predicate is missing, then the node-instance-identifier
         represents all possible server instances for that key.

         This XPath expression is evaluated in the following context:

           o  The set of namespace declarations are those in scope on
              the leaf element where this type is used.

           o  The set of variable bindings contains one variable,
              'USER', which contains the name of user of the current
               session.

           o  The function library is the core function library, but
              note that due to the syntax restrictions of an
              instance-identifier, no functions are allowed.
```

```
            o  The context node is the root node in the data tree.";
     }

     /*
      * Data definition statements
      */

     container nacm {
       nacm:default-deny-all;

       description
         "Parameters for NETCONF Access Control Model.";

       leaf enable-nacm {
         type boolean;
         default true;
         description
           "Enable or disable all NETCONF access control
            enforcement.  If 'true', then enforcement
            is enabled.  If 'false', then enforcement
            is disabled.";
       }

       leaf read-default {
         type action-type;
         default "permit";
         description
           "Controls whether read access is granted if
            no appropriate rule is found for a
            particular read request.";
       }

       leaf write-default {
         type action-type;
         default "deny";
         description
           "Controls whether create, update, or delete access
            is granted if no appropriate rule is found for a
            particular write request.";
       }

       leaf exec-default {
         type action-type;
         default "permit";
         description
           "Controls whether exec access is granted if no appropriate
            rule is found for a particular protocol operation request.";
       }
```

```
        leaf enable-external-groups {
          type boolean;
          default true;
          description
            "Controls whether the server uses the groups reported by the
             NETCONF transport layer when it assigns the user to a set of
             NACM groups.  If this leaf has the value 'false', any group
             names reported by the transport layer are ignored by the
             server.";
        }

        leaf denied-operations {
          type yang:zero-based-counter32;
          config false;
          mandatory true;
          description
            "Number of times a protocol operation request was denied
             since the server last restarted.";
        }

        leaf denied-data-writes {
          type yang:zero-based-counter32;
          config false;
          mandatory true;
          description
            "Number of times a protocol operation request to alter
             a configuration datastore was denied, since the
             server last restarted.";
        }

        leaf denied-notifications {
          type yang:zero-based-counter32;
          config false;
          mandatory true;
          description
            "Number of times a notification was dropped
             for a subscription because access to
             the event type was denied, since the server
             last restarted.";
        }

        container groups {
          description
            "NETCONF Access Control Groups.";

          list group {
            key name;
```

```
          description
            "One NACM Group Entry.  This list will only contain
             configured entries, not any entries learned from
             any transport protocols.";

          leaf name {
            type group-name-type;
            description
              "Group name associated with this entry.";
          }

          leaf-list user-name {
            type user-name-type;
            description
              "Each entry identifies the user name of
               a member of the group associated with
               this entry.";
          }
        }
      }

      list rule-list {
        key "name";
        ordered-by user;
        description
          "An ordered collection of access control rules.";

        leaf name {
          type string {
            length "1..max";
          }
          description
            "Arbitrary name assigned to the rule-list.";
        }
        leaf-list group {
          type union {
            type matchall-string-type;
            type group-name-type;
          }
          description
            "List of administrative groups that will be
             assigned the associated access rights
             defined by the 'rule' list.

             The string '*' indicates that all groups apply to the
             entry.";
        }
```

```
         list rule {
           key "name";
           ordered-by user;
           description
             "One access control rule.

              Rules are processed in user-defined order until a match is
              found.  A rule matches if 'module-name', 'rule-type', and
              'access-operations' matches the request.  If a rule
              matches, the 'action' leaf determines if access is granted
              or not.";

           leaf name {
             type string {
               length "1..max";
             }
             description
               "Arbitrary name assigned to the rule.";
           }

           leaf module-name {
             type union {
               type matchall-string-type;
               type string;
             }
             default "*";
             description
               "Name of the module associated with this rule.

                This leaf matches if it has the value '*', or if the
                object being accessed is defined in the module with the
                specified module name.";
           }
           choice rule-type {
             description
               "This choice matches if all leafs present in the rule
                matches the request.  If no leafs are present, the
                choice matches all requests.";
             case protocol-operation {
               leaf rpc-name {
                 type union {
                   type matchall-string-type;
                   type string;
                 }
                 description
                   "This leaf matches if it has the value '*', or if
                    its value equals the requested protocol operation
                    name.";
```

```
              }
            }
            case notification {
              leaf notification-name {
                type union {
                  type matchall-string-type;
                  type string;
                }
                description
                  "This leaf matches if it has the value '*', or if its
                   value equals the requested notification name.";
              }
            }
            case data-node {
              leaf path {
                type node-instance-identifier;
                mandatory true;
                description
                  "Data Node Instance Identifier associated with the
                   data node controlled by this rule.

                   Configuration data or state data instance
                   identifiers start with a top-level data node.  A
                   complete instance identifier is required for this
                   type of path value.

                   The special value '/' refers to all possible data
                   store contents.";
              }
            }
          }

          leaf access-operations {
            type union {
              type matchall-string-type;
              type access-operations-type;
            }
            default "*";
            description
              "Access operations associated with this rule.

               This leaf matches if it has the value '*', or if the
               bit corresponding to the requested operation is set.";
          }

          leaf action {
            type action-type;
            mandatory true;
```

```
            description
              "The access control action associated with the
               rule.  If a rule is determined to match a
               particular request, then this object is used
               to determine whether to permit or deny the
               request.";
          }

          leaf comment {
            type string;
            description
              "A textual description of the access rule.";
          }
        }
      }
    }
  }


  <CODE ENDS>
```

Figure 5

3.6. IANA Considerations

There are two actions that are requested of IANA: This document registers one URI in "The IETF XML Registry". Following the format in [RFC3688], the following has been registered.

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

This document registers one module in the "YANG Module Names" registry. Following the format in [RFC6020], the following has been registered.

```
     name: ietf-netconf-acm
     namespace: urn:ietf:params:xml:ns:yang:ietf-netconf-acm
     prefix: nacm
     reference: RFC XXXX
        // RFC Ed.: Replace XXX with actual RFC number
        // and remove this note
```

3.7. Security Considerations

This entire document discusses access control requirements and mechanisms for restricting NETCONF protocol behavior within a given session.

This section highlights the issues for an administrator to consider when configuring a NETCONF server with NACM.

3.7.1. NACM Configuration and Monitoring Considerations

Configuration of the access control system is highly sensitive to system security. A server may choose not to allow any user configuration to some portions of it, such as the global security level, or the groups which allowed access to system resources.

By default, NACM enforcement is enabled. By default, "read" access to all datastore contents is enabled, (unless "nacm:default-deny-all" is specified for the data definition) and "exec" access is enabled for safe protocol operations. An administrator needs to ensure that NACM is enabled, and also decide if the default access parameters are set appropriately. Make sure the following data nodes are properly configured:

- /nacm/enable-nacm (default "true")

- /nacm/read-default (default "permit")

- /nacm/write-default (default "deny")

- /nacm/exec-default (default "permit")

An administrator needs to restrict write access to all configurable objects within this data model.

If write access is allowed for configuration of access control rules, then care needs to be taken not to disrupt the access control enforcement. For example, if the NACM access control rules are edited directly within the running configuration datastore (i.e., :writable-running capability is supported and used), then care needs to be taken not to allow unintended access while the edits are being done.

An administrator needs to make sure that the translation from a transport or implementation dependant user identity to a NACM user name is unique and correct. This requirement is specified in detail in section 2.2 of [RFC6241].

An administrator needs to be aware that the YANG data structures representing access control rules (/nacm/rule-list and /nacm/rule-list/rule) are ordered by the client. The server will evaluate the access control rules according to their relative conceptual order within the running datastore configuration.

Note that the /nacm/groups data structure contains the administrative group names used by the server. These group names may be configured locally and/or provided through an external protocol, such as RADIUS [RFC2865] [RFC5607].

An administrator needs to be aware of the security properties of any external protocol used by the NETCONF transport layer to determine group names. For example, if this protocol does not protect against man-in-the-middle attacks, an attacker might be able to inject group names that are configured in NACM, so that a user gets more permissions than it should. In such cases, the administrator may wish to disable the usage of such group names, by setting /nacm/enable-external-groups to "false".

An administrator needs to restrict read access to the following objects within this data model, which reveal access control configuration which could be considered sensitive.

- /nacm/enable-nacm
- /nacm/read-default
- /nacm/write-default
- /nacm/exec-default
- /nacm/enable-external-groups
- /nacm/groups
- /nacm/rule-list

3.7.2. General Configuration Issues

There is a risk that invocation of non-standard protocol operations will have undocumented side effects. An administrator needs to construct access control rules such that the configuration datastore is protected from such side effects.

It is possible for a session with some write access (e.g., allowed to invoke <edit-config>), but without any access to a particular datastore subtree containing sensitive data, to determine the

presence or non-presence of that data. This can be done by repeatedly issuing some sort of edit request (create, update, or delete) and possibly receiving "access-denied" errors in response. These "fishing" attacks can identify the presence or non-presence of specific sensitive data even without the "error-path" field being present within the "rpc-error" response.

It may be possible for the set of NETCONF capabilities on the server to change over time. If so, then there is a risk that new protocol operations, notifications, and/or datastore content have been added to the device. An administrator needs to be sure the access control rules are correct for the new content in this case. Mechanisms to detect NETCONF capability changes on a specific device are outside the scope of this document.

It is possible that the data model definition itself (e.g., YANG when-stmt) will help an unauthorized session determine the presence or even value of sensitive data nodes by examining the presence and values of different data nodes.

There is a risk that non-standard protocol operations, or even the standard <get> protocol operation, may return data which "aliases" or "copies" sensitive data from a different data object. There may simply be multiple data model definitions which expose or even configure the same underlying system instrumentation.

A data model may contain external keys (e.g., YANG leafref), which expose values from a different data structure. An administrator needs to be aware of sensitive data models which contain leafref nodes. This entails finding all the leafref objects that "point" at the sensitive data (i.e., "path-stmt" values) that implicitly or explicitly include the sensitive data node.

It is beyond the scope of this document to define access control enforcement procedures for underlying device instrumentation that may exist to support the NETCONF server operation. An administrator can identify each protocol operation that the server provides, and decide if it needs any access control applied to it.

This document incorporates the optional use of a "recovery session" mechanism, which can be used to bypass access control enforcement in emergencies, such as NACM configuration errors which disable all access to the server. The configuration and identification of such a recovery session mechanism are implementation-specific and outside the scope of this document. An administrator needs to be aware of any "recovery session" mechanisms available on the device, and make sure they are used appropriately.

It is possible for a session to disrupt configuration management, even without any write access to the configuration, by locking the datastore. This may be done to insure all or part of the configuration remains stable while it is being retrieved, or it may be done as a "denial-of-service" attack. There is no way for the server to know the difference. An administrator may wish to restrict "exec" access to the following protocol operations:

- o <lock>
- o <unlock>
- o <partial-lock>
- o <partial-unlock>

3.7.3. Data Model Design Considerations

Designers need to clearly identify any sensitive data, notifications, or protocol operations defined within a YANG module. For such definitions, a "nacm:default-deny-write" or "nacm:default-deny-all" statement ought to be present, in addition to a clear description of the security risks.

Protocol operations need to be properly documented by the data model designer, so it is clear to administrators what data nodes (if any) are affected by the protocol operation, and what information (if any) is returned in the <rpc-reply> message.

Data models ought to be designed so that different access levels for input parameters to protocol operations is not required. Use of generic protocol operations should be avoided, and separate protocol operations defined instead, if different access levels are needed.

4. References

4.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, July 2008.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6021] Schoenwaelder, J., "Common YANG Data Types", RFC 6021, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

4.2. Informative References

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC5607] Nelson, D. and G. Weber, "Remote Authentication Dial-In User Service (RADIUS) Authorization for Network Access Server (NAS) Management", RFC 5607, July 2009.

Appendix A. Usage Examples

The following XML snippets are provided as examples only, to demonstrate how NACM can be configured to perform some access control tasks.

A.1. <groups> Example

There needs to be at least one <group> entry in order for any of the access control rules to be useful.

The following XML shows arbitrary groups, and is not intended to represent any particular use-case.

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
  <groups>
    <group>
      <name>admin</name>
      <user-name>admin</user-name>
      <user-name>andy</user-name>
    </group>

    <group>
      <name>limited</name>
      <user-name>wilma</user-name>
      <user-name>bam-bam</user-name>
    </group>

    <group>
      <name>guest</name>
      <user-name>guest</user-name>
      <user-name>guest@example.com</user-name>
    </group>
  </groups>
</nacm>
```

This example shows 3 groups:

1. The "admin" group contains 2 users named "admin" and "andy".

2. The "limited" group contains 2 users named "wilma" and "bam-bam".

3. The "guest" group contains 2 users named "guest" and "guest@example.com".

A.2. Module Rule Example

Module rules are used to control access to all the content defined in a specific module. A module rule has the <module-name> leaf set, but no case in the "rule-type" choice.

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
  <rule-list>
    <name>guest-acl</name>
    <group>guest</group>

    <rule>
      <name>deny-ncm</name>
      <module-name>ietf-netconf-monitoring</module-name>
      <access-operations>*</access-operations>
      <action>deny</action>
      <comment>
          Do not allow guests any access to the netconf
          monitoring information.
      </comment>
    </rule>
  </rule-list>

  <rule-list>
    <name>limited-acl</name>
    <group>limited</group>

    <rule>
      <name>permit-ncm</name>
      <module-name>ietf-netconf-monitoring</module-name>
      <access-operations>read</access-operations>
      <action>permit</action>
      <comment>
          Allow read access to the netconf
          monitoring information.
      </comment>
    </rule>
    <rule>
      <name>permit-exec</name>
      <module-name>*</module-name>
      <access-operations>exec</access-operations>
      <action>permit</action>
      <comment>
          Allow invocation of the
          supported server operations.
      </comment>
    </rule>
```

```
    </rule-list>

    <rule-list>
      <name>admin-acl</name>
      <group>admin</group>

      <rule>
        <name>permit-all</name>
        <module-name>*</module-name>
        <access-operations>*</access-operations>
        <action>permit</action>
        <comment>
            Allow the admin group complete access to all
            operations and data.
        </comment>
      </rule>
    </rule-list>
  </nacm>
```

This example shows 4 module rules:

deny-ncm: This rule prevents the "guest" group from reading any monitoring information in the "ietf-netconf-monitoring" YANG module.

permit-ncm: This rule allows the "limited" group to read the "ietf-netconf-monitoring" YANG module.

permit-exec: This rule allows the "limited" group to invoke any protocol operation supported by the server.

permit-all: This rule allows the "admin" group complete access to all content in the server. No subsequent rule will match for the "admin" group, because of this module rule.

A.3. RPC Rule Example

RPC rules are used to control access to a specific protocol operation.

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
  <rule-list>
    <name>guest-limited-acl</name>
    <group>limited</group>
    <group>guest</group>

    <rule>
      <name>deny-kill-session</name>
      <module-name>ietf-netconf</module-name>
      <rpc-name>kill-session</rpc-name>
      <access-operations>exec</access-operations>
      <action>deny</action>
      <comment>
        Do not allow the limited or guest group
        to kill another session.
      </comment>
    </rule>
    <rule>
      <name>deny-delete-config</name>
      <module-name>ietf-netconf</module-name>
      <rpc-name>delete-config</rpc-name>
      <access-operations>exec</access-operations>
      <action>deny</action>
      <comment>
        Do not allow limited or guest group
        to delete any configurations.
      </comment>
    </rule>
  </rule-list>

  <rule-list>
    <name>limited-acl</name>
    <group>limited</group>

    <rule>
      <name>permit-edit-config</name>
      <module-name>ietf-netconf</module-name>
      <rpc-name>edit-config</rpc-name>
      <access-operations>exec</access-operations>
      <action>permit</action>
      <comment>
        Allow the limited group to edit the configuration.
      </comment>
    </rule>
  </rule-list>

</nacm>
```

This example shows 3 protocol operation rules:

deny-kill-session: This rule prevents the "limited" or "guest" groups from invoking the NETCONF <kill-session> protocol operation.

deny-delete-config: This rule prevents the "limited" or "guest" groups from invoking the NETCONF <delete-config> protocol operation.

permit-edit-config: This rule allows the "limited" group to invoke the NETCONF <edit-config> protocol operation. This rule will have no real effect unless the "exec-default" leaf is set to "deny".

A.4. Data Rule Example

Data rules are used to control access to specific (config and non-config) data nodes within the NETCONF content provided by the server.

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
  <rule-list>
    <name>guest-acl</name>
    <group>guest</group>

    <rule>
      <name>deny-nacm</name>
      <path xmlns:n="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
        /n:nacm
      </path>
      <access-operations>*</access-operations>
      <action>deny</action>
      <comment>
        Deny the guest group any access to the /nacm data.
      </comment>
    </rule>
  </rule-list>

  <rule-list>
    <name>limited-acl</name>
    <group>limited</group>

    <rule>
      <name>permit-acme-config</name>
      <path xmlns:acme="http://example.com/ns/netconf">
        /acme:acme-netconf/acme:config-parameters
      </path>
      <access-operations>
```

```
          read create update delete
        </access-operations>
        <action>permit</action>
        <comment>
          Allow the limited group complete access to the acme
          netconf configuration parameters.  Showing long form
          of 'access-operations' instead of shorthand.
        </comment>
      </rule>
    </rule-list>

    <rule-list>
      <name>guest-limited-acl</name>
      <group>guest</group>
      <group>limited</group>

      <rule>
        <name>permit-dummy-interface</name>
        <path xmlns:acme="http://example.com/ns/itf">
          /acme:interfaces/acme:interface[acme:name='dummy']
        </path>
        <access-operations>read update</access-operations>
        <action>permit</action>
        <comment>
          Allow the limited and guest groups read
          and update access to the dummy interface.
        </comment>
      </rule>
    </rule-list>

    <rule-list>
      <name>admin-acl</name>
      <group>admin</group>
      <rule>
        <name>permit-interface</name>
        <path xmlns:acme="http://example.com/ns/itf">
          /acme:interfaces/acme:interface
        </path>
        <access-operations>*</access-operations>
        <action>permit</action>
        <comment>
          Allow admin full access to all acme interfaces.
        </comment>
      </rule>
    </rule-list>
  </nacm>
```

This example shows 4 data rules:

deny-nacm: This rule denies the "guest" group any access to the <nacm> subtree. Note that the default namespace is only applicable because this subtree is defined in the same namespace as the <data-rule> element.

permit-acme-config: This rule gives the "limited" group read-write access to the acme <config-parameters>.

permit-dummy-interface: This rule gives the "limited" and "guest" groups read-update access to the acme <interface> entry named "dummy". This entry cannot be created or deleted by these groups, just altered.

permit-interface: This rule gives the "admin" group read-write access to all acme <interface> entries.

A.5. Notification Rule Example

Notification rules are used to control access to a specific notification event type.

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
  <rule-list>
    <name>sys-acl</name>
    <group>limited</group>
    <group>guest</group>

    <rule>
      <name>deny-config-change</name>
      <module-name>acme-system</module-name>
      <notification-name>sys-config-change</notification-name>
      <access-operations>read</access-operations>
      <action>deny</action>
      <comment>
        Do not allow the guest or limited groups
        to receive config change events.
      </comment>
    </rule>
  </rule-list>
</nacm>
```

This example shows 1 notification rule:

deny-config-change: This rule prevents the "limited" or "guest" groups from receiving the acme <sys-config-change> event type.

Appendix B. Change Log

-- RFC Ed.: remove this section before publication.

B.1. 06-07

Added the leaf "enable-external-groups".

Removed dependency to RFC 6242.

Some editorial changes after IESG review.

B.2. 05-06

Added clarification to Security Considerations section about ordered-by user lists (/nacm/rule-list and /nacm/rule-list/rule).

Added clarifications to security considerations wrt/ user names and NETCONF capability changes.

Fixed typos found in review.

B.3. 04-05

Updated Security Considerations section.

Changed term ’operator’ to ’administrator’.

Used the terms "access operation" and "protocol operation" consistently.

Moved some normative text from section 2 to section 3. Also made it more clear that section 2 is not a requirements section, but documentation of the objectives for NACM.

Renamed "nacm:secure" to "nacm:default-deny-write", and "nacm:very-secure" to "nacm:default-deny-all". Explained that "nacm:default-deny-write" is ignored on rpc statements.

Described that <kill-session> and <delete-config> behave as if specified with "nacm:default-deny-all".

B.4. 03-04

Introduced rule-lists to group related rules together.

Moved "module-rule", "rpc-rule", "notification-rule", and "data-rule" into one common "rule", with a choice to select between the four

variants.

Changed "superuser" to "recovery session", and adjusted text throughout document for this change.

Clarified behavior of global default NACM parameters, enable-nacm, read-default, write-default, exec-default.

Clarified when access control is applied during system initialization.

B.5. 02-03

Fixed improper usage of RFC 2119 keywords.

Changed term usage of "database" to "datastore".

Clarified that "secure" and "very-secure" extensions only apply if the /nacm/enable-nacm object is "true".

B.6. 01-02

Removed authentication text and objects.

Changed module name from ietf-nacm to ietf-netconf-acm.

Updated NETCONF and YANG terminology.

Removed open issues section.

Changed some must to MUST in requirements section.

B.7. 00-01

Updated YANG anf YANG Types references.

Updated module namespace URI to standard format.

Updated module header meta-data to standard format.

Filled in IANA section.

B.8. 00

Initial version cloned from draft-bierman-netconf-access-control-02.txt.

Authors' Addresses

Andy Bierman
Brocade

Email: andy@netconfcentral.org

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

NETCONF Working Group
Internet-Draft
Intended status: Experimental
Expires: April 20, 2013

T. Iijima
Hitachi, Ltd.
H. Kimura
Y. Atarashi
H. Higuchi
Alaxala Networks Corp.
Oct 17, 2012

NETCONF over WebSocket
draft-iijima-netconf-websocket-ps-04

Abstract

This memo proposes a way of transporting NETCONF over WebSocket protocol. Web-related technologies are advancing and number of Web-based systems are increasing. Management systems that adopt Web-related technologies or that have browser-based management interface are getting common. It is natural to expect that there is a standard network operation and management protocol to be used over HTTP. Currently, however, there are few efforts in this area. Although NETCONF[RFC6241] once defined itself to be sent over SOAP/HTTPS[RFC4743], supposedly due to unfamiliarity to SOAP or lack of bi-directional capability, such efforts aren’t successful enough[I-D.ietf-netconf-rfc4743-rfc4744-to-historic]. But now, WebSocket protocol, the update of HTTP equipped with bi-directional capability, is available[RFC6455]. This memo describes how NETCONF should be treated over WebSocket protocol.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 20, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
2. Problem Statement 4
3. Use Case 6
4. Concerns about Using HTTP and WebSocket 7
5. Handling of NETCONF Username 8
6. Transporting NETCONF Messages over WebSocket Protocol 9
6.1. Message Sequence 9
6.2. WebSocket Message at Handshake from NETCONF Client 11
6.3. WebSocket Message at Handshake from NETCONF Server 12
6.4. NETCONF Message over WebSocket at NETCONF Client 12
6.5. NETCONF Message over WebSocket at NETCONF Server 14
7. Security Considerations 15
8. IANA Considerations 16
9. Acknowledgements 17
10. References 18
10.1. Normative References 18
10.2. Informative References 18
Authors' Addresses 20

1. Introduction

Web-related technologies are advancing and number of Web-based systems are increasing. Management systems that adopt Web-related technologies or have browser-based management interface are getting common. It is natural to expect that there is a standard network operation and management protocol to be sent over HTTP. Currently, however, there are few efforts in this area and, in most cases, management interface to be used over HTTP are proprietary. Although NETCONF[RFC6241] once defined itself to be sent over SOAP/ HTTPS[RFC4743], supposedly due to unfamiliarity to SOAP or lack of bi-directional capability, such efforts didn’t succeed well enough[I-D.ietf-netconf-rfc4743-rfc4744-to-historic]. But now, WebSocket protocol, the update of HTTP equipped with bi-directional capability, is available[RFC6455]. This memo describes how NETCONF should be exchanged over WebSocket protocol.

This memo does not intend to make WebSocket as a mandatory transport protocol for NETCONF. NETCONF specifies that it is mandatory to be sent over SSH[RFC6241]. But [RFC6241] also specifies in its section 2 that "the NETCONF protocol can be layered on any transport protocol that provides the required set of functionality." According to the specification, those required set of functionality are "Connection-Oriented Operation" and "Authentication, Integrity, and Confidentiality." WebSocket protocol meets those requirements. It is ’connection-oriented.’ And, as written in the section 10.5 of [RFC6455], ’authentication’ is ensured by mechanisms available to a generic HTTP server, such as cookies, HTTP Authentication, or TLS. Moreover, as written in the section 10.6 of [RFC6455], ’integrity and confidentiality’ are ensured by running WebSocket protocol over TLS. For these reasons, WebSocket protocol coupled with TLS meets the requirements of transport protocol for NETCONF.

2. Problem Statement

Web-related technologies, such as JavaScript and JSON(JavaScript Orient Notation), are widely used. And number of Web-based systems, such as those provided for IaaS (Infrastructure as a Service), are increasing. There are systems that are providing management interface in a form of REST API(Application Programming Interface). It is natural to expect that there is a standard network operation and management protocol to be sent over HTTP. Currently, however, there are few efforts in this area. And, in most of the cases, management interface are proprietary.

NETCONF[RFC6241] once defined itself to be sent over SOAP/ HTTPS[RFC4743], as drawn in Figure 1. But, supposedly due to unfamiliarity to SOAP or lack of bi-directional capability, such efforts haven't been successful enough[I-D.ietf-netconf-rfc4743-rfc4744-to-historic].

```
          Layer                 Example
     +-------------+      +-----------------+      +----------------+
(4)  |   Content   |      |  Configuration  |      |  Notification  |
     |             |      |      data       |      |      data      |
     +-------------+      +-----------------+      +----------------+
            |                     |                       |
     +-------------+      +-----------------+              |
(3)  | Operations  |      |  <edit-config>  |              |
     |             |      |                 |              |
     +-------------+      +-----------------+              |
            |                     |                       |
     +-------------+      +-----------------+      +----------------+
(2)  |  Messages   |      |     <rpc>,      |      | <notification> |
     |             |      |   <rpc-reply>   |      |                |
     +-------------+      +-----------------+      +----------------+
            |                     |                       |
     +-------------+      +----------------------------------------+
(1)  |   Secure    |      |  SSH, TLS, BEEP/TLS, SOAP/HTTP/TLS, ... |
     |  Transport  |      |                                        |
     +-------------+      +----------------------------------------+
```

Figure 1: NETCONF Protocol Layers

As of now, however, WebSocket protocol is available[RFC6455]. It is based on HTTP, which is familiar to all. And, it has a bi-directional capability. Thus, by using WebSocket protocol, it's possible to realize NETCONF that is used over HTTP with ease and that supports notification mechanism specified in [RFC5277].

Some WebSocket implementations already exist today. As examples of WebSocket server implementation, Jetty[Jetty], Kaazing[Kaazing], and the like are available. And, as examples of WebSocket client implementation, Jetty[Jetty] is available. Moreover, as examples of Web-browsers that can work as WebSocket client, Chrome[Chrome], and FireFox[FireFox] and the like are available.

WebSocket server and client implementations mentioned above are providing libraries. And WebSocket protocol itself also defines API[WebSocket API] to be used on Web-browsers. Thus, by using those libraries and APIs, developers can develop NETCONF server, install-based NETCONF client, and browser-based NETCONF client with ease that use WebSocket for transporting NETCONF.

3. Use Case

XML, which NETCONF uses for message encoding, has high compatibility with HTTP in that XML are easily manipulated on Web-browsers by JavaScript DOM (Document Object Model) API. Hence, there are cases in which XML is used over HTTP to manage network devices. But as far as those XML are proprietary, the way of managing network devices and items to manage are different from network device to network device. If NETCONF, instead of proprietary XML, and its data models are used for above cases, it will provide a way of managing various network devices in a same manner.

Browser-based network management systems don’t require installation on computers. For this reason, some operators prefer browser-based network management systems. This trend might accelerate in the age of tablet computers. In this respect, it is rational for NETCONF to have an option to be used through browser-based network management system. Operators will be able to manage network devices from any computers without installation.

4. Concerns about Using HTTP and WebSocket

There are some drawbacks inherent in HTTP as mentioned in the section 2.4 of [RFC4743], which describes the way of transporting NETCONF over SOAP/HTTPS. But, for these drawbacks, the same section makes suggestions. Those suggestions are effective when WebSocket is used for transporting NETCONF. That is, intermediate proxies SHOULD not be used since it may close idle connections. And, the fields of 'Cache-Control' and 'Pragma' in HTTP header, which is sent before or during WebSocket opening handshake, SHOULD be specified as 'no-cache.'

WebSocket has had several security concerns. But it incorporated its own security mechanisms such as origin header and masking, as stated in the Security Considerations section of [RFC6455]. In any case, using TLS is necessary for ensuring authentication and confidentiality, when WebSocket is used for transporting NETCONF.

5. Handling of NETCONF Username

NETCONF[RFC6241] mandates underlying transport protocol to carry NETCONF username and to provide it to NETCONF server. In the case of transporting NETCONF over WebSocket, this memo proposes that NETCONF username SHOULD be carried and provided in compliance with NETCONF over TLS[I-D.badra-netconf-rfc5539bis].

In the case of transporting NETCONF over WebSocket, TLS is necessary for the user authentication. In order to ensure that NETCONF access control is done consistently with TLS authentication, NETCONF username SHOULD be matched with TLS authentication data. At present, [I-D.badra-netconf-rfc5539bis] is proposing ways of extracting NETCONF username from TLS authentication data. Being compliant with these approaches is the most efficient. At least, it is confirmed that NETCONF server using WebSocket/TLS for underlying protocol can see TLS Certificate at the time of TLS handshake and can extract NETCONF username from the Certificate.

6. Transporting NETCONF Messages over WebSocket Protocol

This section specifies how NETCONF messages are exchanged between NETCONF client and server over WebSocket protocol.

6.1. Message Sequence

Simplified overall message sequence is depicted in Figure 2. This sequence is depicting the case in which NETCONF client runs on a Web-browser. However, it must be noted that there is also a case in which NETCONF client runs as an install-based software developed with WebSocket client library.

```
+---------------------------+  +---------------------------+
| Network Management System |  |     Network Device        |
|                           |  |   (e.g. 192.168.1.1)      |
| +---------++-----------+  |  | +-----------++---------+  |
| | NETCONF || WebSocket |  |  | | WebSocket || NETCONF |  |
| | Client  ||  (HTTP)   |  |  | |  (HTTP)   || Server  |  |
| |         ||  Client   |  |  | |  Server   ||         |  |
| +---------++-----------+  |  | +-----------++---------+  |
+------|-----------|--------+  +--------|-----------|------+
       |           |                    |           |
       |Load *.html|                    |           |
       |---------->|     Hello(TLS)     |           |
       |     /     |------------------->|           |
       |    |      | Server Certificate |           |
       |    TLS    |<-------------------|           |
       | handshake | Client Certificate |           |
       |    |      |------------------->|NETCONF username
       |    |      |   Finished(TLS)    |---------->|
       |     \     |<-------------------|           |
       |           |                    |           |
       |           |     GET(HTTP)      |           |
       |           |------------------->|           |
       |           |    200 OK(HTTP)    |           |
       |           |<-------------------|           |
       |<----------|                    |           |
                              ....
       |Call WebSocket() API            |           |
       |---------->|                    |           |
       |           |Connection: Upgrade |           |
       |     /     |------------------->|           |
       | WebSocket |Connection: Upgrade |           |
       | opening   |<-------------------|           |
       | handshake |     ACK(TCP)       |           |
       |     \     |------------------->|           |
```

```
          |Invoke onopen = function() { }  |           |
          |<----------|                    |           |
          |           |                    |           |
          |<hello>(NETCONF)                |           |
          |---------->|                    |           |
          |           |------------------->|           |
          |           |                    |---------->|
          |           |                    |  <hello>  |
          |           |                    |<----------|
          |           |<-------------------|           |
          |<----------|                    |           |
                                  ....
          |<create-subscription>(NETCONF)  |           |
          |---------->|                    |           |
          |           |------------------->|           |
          |           |                    |---------->|
          |           |                    |   <ok>    |
          |           |                    |<----------|
          |           |<-------------------|           |
          |<----------|                    |           |
                                  ....
          |           |        <notification>(NETCONF)|
          |           |                    |<----------|
          |           |<-------------------|           |
          |<----------|                    |           |
          |           |                    |           |
```

Figure 2: Message Sequence

First of all, a browser starts loading of an html file, which imports the code of NETCONF client, over TLS. This invokes TLS handshake. At the time of TLS handshake, NETCONF server can extract NETCONF username from Certificate according to the algorithm described in [I-D.badra-netconf-rfc5539bis]. After TLS handshake is complete, the html file is loaded onto the browser.

The loaded html file works as NETCONF client and it initiates WebSocket opening handshake. When NETCONF server receives a WebSocket connection request, it notifies the client of whether WebSocket handshake has succeeded. After WebSocket connection is established, NETCONF client starts sending NETCONF messages over the connection.

NETCONF <hello> messages are exchanged between NETCONF client and server, at first, so that a NETCONF session is established and a NETCONF session ID is allocated by NETCONF server to the NETCONF client. Then, NETCONF <rpc> messages are exchanged. After NETCONF

<rpc> message of <create-subscription> request is approved by the NETCONF server, NETCONF <notification> messages are sent from NETCONF server asynchronously.

When WebSocket server shuts down, NETCONF session as well as WebSocket connection is killed. Since NETCONF notification subscription is associated with NETCONF session ID as written in section 3.5 of NETCONF notification mechanism[RFC5277], subscription status is lost when WebSocket server shuts down. Thus, it is necessary to redo WebSocket opening handshake, NETCONF session establishment, and NETCONF notification subscription after WebSocket server reboots.

Without any indications, TCP port numbers of 443 is automatically used for transporting NETCONF messages over WebSocket over TLS. When port number other than 443 needs to be used, the number SHOULD be specified at both NETCONF client and server respectively.

6.2. WebSocket Message at Handshake from NETCONF Client

WebSocket opening handshake is necessary for the establishment of an WebSocket connection, which is used for NETCONF message exchange. The WebSocket handshake is initiated by a NETCONF client. Figure 3 is an example of WebSocket message sent from the NETCONF client at the time of WebSocket handshake.

```
C: GET /netconf HTTP/1.1
C: Host: 192.168.1.1
C: Upgrade: websocket
C: Connection: Upgrade
C: Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
C: Origin: http://192.168.1.1
C: Sec-WebSocket-Protocol: netconf
C: Sec-WebSocket-Version: 13
```

Figure 3: WebSocket Message from NETCONF Client

Most of the fileds in Figure 3 are generated automatically by WebSocket client. The only field that the NETCONF client needs to specify is ’Sec-WebSocket-Protocol’ field, so-called subprotocol field. The NETCONF client has to specify the field as ’netconf,’ for example, so that the NETCONF server understands that messages sent over this connection is directed towards NETCONF server.

Aforementioned establishment of the WebSocket connection and specification of subprotocol is made by using WebSocket API in the

html file of the NETCONF client as depicted in Figure 4.

```
var wssURL = "wss://" + location.host;
var wss = new WebSocket(wssURL, "netconf");
```

Figure 4: WebSocket API in NETCONF Clients for Initiating Handshake

6.3. WebSocket Message at Handshake from NETCONF Server

Figure 5 is an example of WebSocket message sent from the NETCONF server to the NETCONF client at the time of WebSocket handshake.

```
S: HTTP/1.1 101 Switching Protocols
S: Upgrade: websocket
S: Connection: Upgrade
S: Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
S: Sec-WebSocket-Protocol: netconf
```

Figure 5: WebSocket Message from NETCONF Server

Most of the fields in Figure 5 are generated automatically by WebSocket server, too. The only field that the NETCONF server needs to specify is ’Sec-WebSocket-Protocol’ field. The NETCONF server has to specify the field as ’netconf,’ for example, in order to let the NETCONF client know that the WebSocket server in the network device accepts NETCONF messages.

Unlike the WebSocket client, there’s no standardized WebSocket API for WebSocket server. But, there are some WebSocket server implementations as mentioned in section 2. Thus, it’s possible to develop a part of WebSocket opening handshake at NETCONF server with ease by using libraries provided by those implementations.

6.4. NETCONF Message over WebSocket at NETCONF Client

NETCONF message exchange between NETCONF client and server starts after an WebSocket connection is established. <hello> messages are exchanged, first, for the establishment of a NETCONF session and allocation of NETCONF session ID. Then, <rpc> messages like <edit-config> are sent from NETCONF client to NETCONF server for NETCONF configurations. And <rpc> messages like <create-subscription> are sent from NETCONF client to NETCONF server for subscription of NETCONF notification. Moreover, messages like <notification> are sent asynchronously from NETCONF server to NETCONF client for NETCONF

notification. During this data transfer period, both NETCONF configuration messages and NETCONF notification messages are encapsulated as a payload of WebSocket protocol according to the Data Framing specified in the section 5.2 of WebSocket protocol[RFC6455]. This encapsulation is made by WebSocket layer.

Sending and receiving of NETCONF messages at NETCONF client is made by using WebSocket API. The example is illustrated in Figure 6.

```
wss.onopen = function() {
  // WebSocket connection has been established.
  // Thus, NETCONF <hello> can be sent here
  // by send() method.
  wss.send("message to send");
  ....
};

wss.onmessage = function (evt) {
  // NETCONF message has arrived.
  // Thus, NETCONF message can be parsed here by DOM APIs.

  var parser = new DOMParser();
  var dom    = parser.parseFromString(evt.data, "text/xml");
  ....
  if(dom.documentElement.nodeName == "hello"){
    // The NETCONF message turned out to be
    // NETCONF <hello>.
    // Subsequent NETCONF message can be sent here
    // by send() method.
    wss.send("message to send");
    ...
  }
};
```

Figure 6: WebSocket API in NETCONF Client for Sending Messages

As shown in Figure 6, NETCONF messages are sent from NETCONF client by using WebSocket API of "wss.send()." And, NETCONF messages are received at NETCONF client by using WebSocket API of "wss.onmessage()."

The contents of NETCONF messages exchanged through above API might be either a hand-written XML messages typed into network management system or XML messages created by JavaScript DOM API according to the data set on the network management system. It must be noted that in the case of transporting NETCONF over WebSocket, <rpc> parts need to

be created by NETCONF client, unlike the case of transporting NETCONF over SOAP/HTTPS, in which <rpc> parts are generated in SOAP layer.

6.5. NETCONF Message over WebSocket at NETCONF Server

Unlike the WebSocket client, the way of sending and receiving NETCONF message at NETCONF server is proprietary since there’s no standardized API for WebSocket server. But, there are some WebSocket server implementations as mentioned in section 2. Thus, it’s possible to develop NETCONF server with ease by using libraries provided by those implementations.

7. Security Considerations

It is necessary to use TLS (Transport Layer Security) in order to ensure Transport-level security, such as authentication of users and encryption of data transfer. That is, NETCONF has to be sent in the form of NETCONF over WebSocket over TLS (WSS).

In addition, the security considerations of NETCONF protocol[RFC6241], NETCONF Notification mechanism[RFC5277], and WebSocket protocol[RFC6455] are applicable to this document. Implementers or users SHOULD take these considerations into account.

8. IANA Considerations

As written in section 11.5 of WebSokcet protocol[RFC6455], NETCONF's Subprotocol Common Name, to be exchanged in 'Sec-WebSocket-Protocol' field at WebSocket opening handshake, coupled with Identifier and Definition need to be registered with the WebSocket Subprotocol Name Registry.

9. Acknowledgements

This document was written using the xml2rfc tool described in [RFC2629].

10. References

10.1. Normative References

[I-D.badra-netconf-rfc5539bis]
Badra, M., "NETCONF Over Transport Layer Security (TLS)", draft-badra-netconf-rfc5539bis-02 (work in progress), April 2012.

[I-D.ietf-netconf-rfc4743-rfc4744-to-historic]
Wijnen, B., "RFC4743 and RFC4744 to Historic status", draft-ietf-netconf-rfc4743-rfc4744-to-historic-00 (work in progress), September 2012.

[RFC4743] Goddard, T., "Using NETCONF over the Simple Object Access Protocol (SOAP)", RFC 4743, December 2006.

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, July 2008.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, December 2011.

[WebSocket API]
"The WebSocket API".

<http://dev.w3.org/html5/websockets/>

10.2. Informative References

[Chrome] "Chrome".

<http://www.google.com/chrome/?hl=en>

[FireFox] "FireFox".

<http://www.mozilla.org/>

[Jetty] "Jetty WebServer".

<http://jetty.codehaus.org/jetty/>

[Kaazing] "Kaazing".

<http://kaazing.com/>

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

Authors' Addresses

Tomoyuki Iijima
Hitachi, Ltd.
292 Yoshida-cho, Totsuka-ku
Yokohama, Kanagawa 244-0817
Japan

Phone: +81-45-860-2156
Email: tomoyuki.iijima.fg@hitachi.com

Hiroyasu Kimura
Alaxala Networks Corp.
Shin-Kawasaki Mitsui Bldg.
890 Saiwai-ku Kashimada
Kawasaki, Kanagawa 212-0058
Japan

Phone: +81-44-549-1735
Fax: +81-44-549-1272
Email: h-kimura@alaxala.net

Yoshifumi Atarashi
Alaxala Networks Corp.
Shin-Kawasaki Mitsui Bldg.
890 Saiwai-ku Kashimada
Kawasaki, Kanagawa 212-0058
Japan

Phone: +81-44-549-1735
Fax: +81-44-549-1272
Email: atarashi@alaxala.net

Hidemitsu Higuchi
Alaxala Networks Corp.
Shin-Kawasaki Mitsui Bldg.
890 Saiwai-ku Kashimada
Kawasaki, Kanagawa 212-0058
Japan

Phone: +81-44-549-1735
Fax: +81-44-549-1272
Email: hidemitsu.higuchi@alaxala.com