NETCONF Working Group
Internet-Draft
Obsoletes: 5539 (if approved)
Intended status: Standards Track
Expires: March 17, 2013

M. Badra
LIMOS Laboratory

September 13, 2012

NETCONF Over Transport Layer Security (TLS)
draft-ietf-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 March 17, 2013.

Copyright Notice

Table of Contents

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. The client actively opens the TLS connection and the server 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 used the same framing sequence defined in [RFC6242], 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

Exiting NETCONF is accomplished using the <close-session> operation. A NETCONF server will process NETCONF messages from the NETCONF client in the order in which they are received. When the NETCONF server processes a <close-session> operation, the NETCONF server SHALL respond and close the TLS session channel. The NETCONF server MUST NOT process any NETCONF messages received after the <close-session> operation. The TLS session is closed as described in [RFC6242] Section 7.2.1.

3. Endpoint Authentication, Identification and Authorization

3.1. Server Identity

If the server's presented certificate has passed certification path validation [RFC5280] to a configured trust anchor, 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 and guidelines defined in [RFC6125].

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]. The NETCONF server MUST implement the algorithms for deriving NETCONF usernames from presented certificates that are documented in the ietf-netconf-tls YANG module. This YANG module lets the NETCONF security administrator configure how the NETCONF server derives NETCONF usernames from presented certificates. It also lets different certificate-to-username derivation algorithms be used for different certificates.

When a NETCONF server accepts a TLS connection from a NETCONF client, the NETCONF server attempts to derive a NETCONF username from the certificate presented by the NETCONF client. If the NETCONF server cannot derive a valid NETCONF username from the client's presented certificate, then the NETCONF server MUST close the TLS connection, and MUST NOT accept NETCONF messages over it. The NETCONF server MAY use any of the following algorithms to produce the NETCONF username from the certificate presented by the NETCONF client:

- Map a certificate directly to a specified, pre-configured, NETCONF username;

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

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

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

- Examine the subjectAltName's rfc822Name, dnsName, and iPAddress fields in a pre-defined order. Return the value from the first subjectAltName field that is examined, defined, and populated with a non-empty value. If no subjectAltName field of a specific type is defined, then the examination skips that field and proceeds to examine the next field type. If a subjectAltName field is defined, but the value is not populated, or is populated by an empty value, then the examination skips that field and proceeds to examine the next field type.

The NETCONF server MUST implement all of these algorithms, and allow the deployer to choose the algorithm used. The certificate-to-username-transforms container in the ietf-netconf-tls YANG module specifies how a NETCONF server transforms a 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, a fingerprint can be a safe short-hand for identifying a certificate.

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. 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 look up the key corresponding to the presented PSK identity. If the selected pre-shared keys match and the key is valid, then the client is authenticated and the NETCONF username associated with the PSK identity. For details on how the PSK identity MAY be encoded in UTF-8, see section 5.1. of RFC [RFC6241].

3.2.1.3. Remote Configuration

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

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

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

 prefix "nctls";

 import ietf-yang-types {
  prefix yang;
 }

 import ietf-netconf-acm {
  prefix nacm;
 }

 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 servers transform
   pre-shared TLS keys into NETCONF usernames.

   This YANG module is patterned after 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) 2012 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 "2012-09-13" {
   description
     "rename host-part to domain-part in the description of RFC822".
   reference
     "RFC XXXX: NETCONF over Transport Layer Security (TLS)";
 }

 revision "2012-04-27" {
   description
     "Changed data type of fingerprints and keys from binary to string.";
   reference
     "RFC XXXX: NETCONF over Transport Layer Security (TLS)";
 }

 revision "2012-04-11" {
  description
    "Incorporates comments on draft-badra-netconf-rfc5539bis-01.txt.";
  reference
    "RFC XXXX: NETCONF over Transport Layer Security (TLS)";
 }

 revision "2012-02-13" {
  description
    "Initial version";
  reference
    "RFC XXXX: NETCONF over Transport Layer Security (TLS)";
 }


 feature map-certificates {
  description
    "The :map-certificates capability implements mapping X.509
     certificates to NETCONF user names.";
 }


 feature map-pre-shared-keys {
  description
    "The :map-pre-shared-keys capability implements mapping TLS
     pre-shared keys to NETCONF user names.";
 }
```

```
 typedef tls-fingerprint-type {
  type string {
    pattern '([0-9a-fA-F]){2}(:([0-9a-fA-F]){2})*';
  }
  description
    "A cryptographic signature (fingerprint) value that can be used to
     uniquely reference other data of potentially arbitrary length.";
 }


 //
 //  Objects related to deriving NETCONF usernames from X.509
 //  certificates.
 //

 container cert-mappings {
  if-feature map-certificates;
  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 certificate chain validation is expected to occur through an
     out-of-band transfer.

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

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

         2) If the list entry's fingerprint value matches that
            of a locally held copy of a trusted CA certificate, and
            that CA certificate was part of the CA certificate chain
            to the presented certificate, then consider the list entry
            as a successful match.
```

```
            This feature lets the NETCONF server derive NETCONF
            usernames from all certificates signed by the trusted
            CA certificate.  The NETCONF server will derive all
            NETCONF usernames using the same derivation algorithm.
            The NETCONF server requires only a single list entry
            to configure this behavior.

      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 leaf'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, then the
      NETCONF server MUST close the connection, and MUST NOT accept
      NETCONF messages over it.

      Non-consecutive 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).

      Security administrators 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.";


  list cert-map {
    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.";
    }

    container fingerprint {
      choice algorithm-and-hash {
        leaf md5 {
          type tls-fingerprint-type;
        }
        leaf sha1 {
          type tls-fingerprint-type;
        }
        leaf sha224 {
          type tls-fingerprint-type;
        }
        leaf sha256 {
          type tls-fingerprint-type;
        }
        leaf sha384 {
          type tls-fingerprint-type;
        }
        leaf sha512 {
          type tls-fingerprint-type;
        }
        description
          "Specifies the signature algorithm and cryptographic
           signature (fingerprint) used to identify an X.509
           certificate.

           Implementations of this YANG module MAY, but are not
           required to, implement all of these cryptographic signature
           algorithms.  Implementations of this YANG module MUST
           implement at least one of these cryptographic signature
           algorithms.

           The available choices may be extended in the future as
           stronger cryptographic signature algorithms become
           available and are deemed necessary.";

        reference
          "RFC 5246: The Transport Layer Security (TLS) Protocol
           Version 1.2; Section 7.4.1.4.1,  Signature Algorithms";
      }  // choice algorithm-and-hash
    }    // container fingerprint
```

```
    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;  }
      }

      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 subsequent
        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

          The NETCONF server derives the NETCONF username from the
          subjectAltName fields rfc822Name, dnsName, and ipAddress,
          as described in sections above.  This enumeration specifies
          the NETCONF server first examines the rfc822Name, then
          examines the dnsName, then finally examines the ipAddress.
          The first matching subjectAltName value found in the
          certificate MUST be used when deriving the NETCONF username.

          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 leaf.  Only some
        mapping systems will make use of this leaf.  When the NETCONF
        server derives the NETCONF username from the client's presented
        certificate, the value in this leaf MUST be ignored for any
        mapping type that does not require data present in this leaf.";
    }
  }  // list cert-map
 }    // container cert-mappings


 //
 //  Objects related to deriving NETCONF usernames from TLS pre-shared
 //  keys.
 //

 container psk-map {
  if-feature map-pre-shared-keys;
  list psk {
    key psk-identity;

    leaf psk-identity {
      type string;
      description
        "The PSK identity encoded as a UTF-8 string.";
      reference
        "RFC 4279: Pre-Shared Key Ciphersuites for Transport Layer
                   Security (TLS)";
    }

    leaf user-name {
      type nacm:user-name-type;
      mandatory true;
      description
       "The NETCONF username associated with this PSK identity.";
    }

    leaf valid-not-before {
      type yang:date-and-time;
      description
        "This PSK identity is not valid before the given data
         and time.";
    }

    leaf valid-not-after {
      type yang:date-and-time;
      description
        "This PSK identity is not valid before the given data
         and time.";
```

```
    }

    leaf key {
      type string;
      pattern '([0-9a-fA-F]){2}(:([0-9a-fA-F]){2})*';
      nacm:default-deny-all;
      description
        "The key associated with the PSK identity";
    }
  }
 }    // container psk-map
}
```

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, 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].

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, mbadra@gmail.com.
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 previous versions 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

Juergen Schoenwaelder
Jacobs University Bremen
Campus Ring 1
28725 Bremen
Germany

EMail: j.schoenwaelder@jacobs-university.de

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.

[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.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, March 2011.

[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.

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

[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 draft-badra-netconf-rfc5539bis-02 to draft-ietf-netconf-rfc5539bis-00

- Remove the reference to BEEP
- rename host-part to domain-part in the description of RFC822.

Author's Address

Mohamad Badra
LIMOS Laboratory

Email: mbadra@gmail.com