Internet Engineering Task Force (IETF)
Request for Comments: 6242
Obsoletes: 4742
Category: Standards Track
ISSN: 2070-1721

M. Wasserman
Painless Security, LLC

June 2011

Using the NETCONF Protocol over Secure Shell (SSH)

Abstract

This document describes a method for invoking and running the Network Configuration Protocol (NETCONF) within a Secure Shell (SSH) session as an SSH subsystem. This document obsoletes RFC 4742.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6242.

Table of Contents

1. Introduction

The NETCONF protocol [RFC6241] is an XML-based protocol used to manage the configuration of networking equipment. NETCONF is defined to be session-layer and transport independent, allowing mappings to be defined for multiple session-layer or transport protocols. This document defines how NETCONF can be used within a Secure Shell (SSH) session, using the SSH connection protocol [RFC4254] over the SSH transport protocol [RFC4253]. This mapping will allow NETCONF to be executed from a secure shell session by a user or application.

Although this document gives specific examples of how NETCONF messages are sent over an SSH connection, use of this transport is not restricted to the messages shown in the examples below. This transport can be used for any NETCONF message.

2. Requirements 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 RFC 2119 [RFC2119].

3. Starting NETCONF over SSH

To run NETCONF over SSH, the SSH client will first establish an SSH transport connection using the SSH transport protocol, and the SSH client and SSH server will exchange keys for message integrity and encryption. The SSH client will then invoke the "ssh-userauth" service to authenticate the user, as described in the SSH

authentication protocol [RFC4252]. Once the user has been successfully authenticated, the SSH client will invoke the "ssh-connection" service, also known as the SSH connection protocol.

The username provided by the SSH 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 SSH session MUST be dropped. Any transformations applied to the authenticated identity of the SSH client made by the SSH server (e.g., via authentication services or mappings to system accounts) are outside the scope of this document.

After the ssh-connection service is established, the SSH client will open a channel of type "session", which will result in an SSH session.

Once the SSH session has been established, the NETCONF client will invoke NETCONF as an SSH subsystem called "netconf". Subsystem support is a feature of SSH version 2 (SSHv2) and is not included in SSHv1. Running NETCONF as an SSH subsystem avoids the need for the script to recognize shell prompts or skip over extraneous information, such as a system message that is sent at shell start-up.

In order to allow NETCONF traffic to be easily identified and filtered by firewalls and other network devices, NETCONF servers MUST default to providing access to the "netconf" SSH subsystem only when the SSH session is established using the IANA-assigned TCP port 830. Servers SHOULD be configurable to allow access to the netconf SSH subsystem over other ports.

A user (or application) could use the following command line to invoke NETCONF as an SSH subsystem on the IANA-assigned port:

```
[user@client]$ ssh -s server.example.org -p 830 netconf
```

Note that the -s option causes the command ("netconf") to be invoked as an SSH subsystem.

3.1. Capabilities Exchange

As specified in [RFC6241], the NETCONF server indicates its capabilities by sending an XML document containing a <hello> element as soon as the NETCONF session is established. The NETCONF client can parse this message to determine which NETCONF capabilities are supported by the NETCONF server.

As [RFC6241] states, the NETCONF client also sends an XML document containing a <hello> element to indicate the NETCONF client's capabilities to the NETCONF server. The document containing the <hello> element is the first XML document that the NETCONF client sends after the NETCONF session is established.

The following example shows a capability exchange. Data sent by the NETCONF client are marked with "C:", and data sent by the NETCONF server are marked with "S:".

```
S: <?xml version="1.0" encoding="UTF-8"?>
S: <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
S:   <capabilities>
S:     <capability>
S:       urn:ietf:params:netconf:base:1.1
S:     </capability>
S:     <capability>
S:       urn:ietf:params:ns:netconf:capability:startup:1.0
S:     </capability>
S:   </capabilities>
S:   <session-id>4</session-id>
S: </hello>
S: ]]>]]>

C: <?xml version="1.0" encoding="UTF-8"?>
C: <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
C:   <capabilities>
C:     <capability>
C:       urn:ietf:params:netconf:base:1.1
C:     </capability>
C:   </capabilities>
C: </hello>
C: ]]>]]>
```

Although the example shows the NETCONF server sending a <hello> message followed by the NETCONF client's <hello> message, both sides will send the message as soon as the NETCONF subsystem is initialized, perhaps simultaneously.

4. Using NETCONF over SSH

A NETCONF over SSH session consists of a NETCONF client and NETCONF server exchanging complete XML documents. Once the session has been established and capabilities have been exchanged, the NETCONF client will send complete XML documents containing <rpc> elements to the server, and the NETCONF server will respond with complete XML documents containing <rpc-reply> elements.

4.1. Framing Protocol

The previous version of this document defined the character sequence "]]>]]>" as a message separator, under the assumption that it could not be found in well-formed XML documents. However, this assumption is not correct. It can legally appear in XML attributes, comments, and processing instructions. In order to solve this problem, and at the same time be compatible with existing implementations, this document defines the following framing protocol.

The <hello> message MUST be followed by the character sequence]]>]]>. Upon reception of the <hello> message, the receiving peer's SSH 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 (see Section 4.2) is used for the remainder of the NETCONF session. Otherwise, the old end-of-message-based mechanism (see Section 4.3) is used.

4.2. Chunked Framing Mechanism

This mechanism encodes all NETCONF messages with a chunked framing. Specifically, the message follows the ABNF [RFC5234] rule Chunked-Message:

```
Chunked-Message = 1*chunk
                  end-of-chunks

chunk           = LF HASH chunk-size LF
                  chunk-data
chunk-size      = 1*DIGIT1 0*DIGIT
chunk-data      = 1*OCTET

end-of-chunks   = LF HASH HASH LF

DIGIT1          = %x31-39
DIGIT           = %x30-39
HASH            = %x23
LF              = %x0A
OCTET           = %x00-FF
```

The chunk-size field is a string of decimal digits indicating the number of octets in chunk-data. Leading zeros are prohibited, and the maximum allowed chunk-size value is 4294967295.

As an example, the message:

```
    <rpc message-id="102"
         xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <close-session/>
    </rpc>
```

could be encoded as (using '\n' as a visible representation of the LineFeed character):

```
C:  \n#4\n
C:  <rpc
C:  \n#18\n
C:   message-id="102"\n
C:  \n#79\n
C:       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">\n
C:    <close-session/>\n
C:  </rpc>
C:  \n##\n
```

Conceptually, the SSH Transport layer encodes messages sent by the Messages layer, and decodes messages received on the SSH channel before passing them to the Messages layer.

The examples for the chunked framing mechanism show all LineFeeds, even those that are not used as part of the framing mechanism. Note that the SSH transport does not interpret the XML content; thus, it does not care about any optional XML-specific LineFeeds.

In the second and third chunks quoted above, each line is terminated by a LineFeed. For all the XML lines (except the last one), this example treats the LineFeed as part of the chunk-data and so contributing to the chunk-size.

Note that there is no LineFeed character after the <rpc> end tag in this message. The LineFeed required by the start of the end-of-chunks block immediately follows the last '>' character in the message.

If the chunk-size and the chunk-size value respectively are invalid or if an error occurs during the decoding process, the peer MUST terminate the NETCONF session by closing the corresponding SSH channel. Implementations MUST ensure they are not vulnerable for a buffer overrun.

4.3. End-of-Message Framing Mechanism

This mechanism exists for backwards compatibility with implementations of previous versions of this document. It is only used when the remote peer does not advertise a base protocol version supporting chunked encoding, i.e., a NETCONF implementation only supporting :base:1.0.

When this mechanism is used, the special character sequence]]>]]>, MUST be sent by both the NETCONF client and the NETCONF server after each message (XML document) in the NETCONF exchange. Conceptually, the SSH Transport layer passes any data found in between the]]>]]> characters to the Messages layer.

A NETCONF over SSH session, using the backwards-compatible end-of-message framing to retrieve a set of configuration information, might look like this:

```
C: <?xml version="1.0" encoding="UTF-8"?>
C: <rpc message-id="105"
C: xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
C:   <get-config>
C:     <source><running/></source>
C:     <config xmlns="http://example.com/schema/1.2/config">
C:      <users/>
C:     </config>
C:   </get-config>
C: </rpc>
C: ]]>]]>

S: <?xml version="1.0" encoding="UTF-8"?>
S: <rpc-reply message-id="105"
S: xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
S:   <config xmlns="http://example.com/schema/1.2/config">
S:     <users>
S:       <user><name>root</name><type>superuser</type></user>
S:       <user><name>fred</name><type>admin</type></user>
S:       <user><name>barney</name><type>admin</type></user>
S:     </users>
S:   </config>
S: </rpc-reply>
S: ]]>]]>
```

5. Exiting the NETCONF Subsystem

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 SSH session channel. The NETCONF server MUST NOT process any NETCONF messages received after the <close-session> operation.

To continue the example used in Section 4.2, an existing NETCONF subsystem session could be closed as follows:

```
C: \n#140\n
C: <?xml version="1.0" encoding="UTF-8"?>\n
C: <rpc message-id="106"\n
C:      xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">\n
C:   <close-session/>\n
C: </rpc>
C: \n##\n

S: \n#139\n
S: <?xml version="1.0" encoding="UTF-8"?>\n
S: <rpc-reply id="106"\n
S:            xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">\n
S:   <ok/>\n
S: </rpc-reply>
S: \n##\n
```

6. Security Considerations

NETCONF is used to access configuration and state information and to modify configuration information, so the ability to access this protocol should be limited to users and systems that are authorized to view the NETCONF server’s configuration and state or to modify the NETCONF server’s configuration.

The identity of the SSH server MUST be verified and authenticated by the SSH client according to local policy before password-based authentication data or any configuration or state data is sent to or received from the SSH server. The identity of the SSH client MUST also be verified and authenticated by the SSH server according to local policy to ensure that the incoming SSH client request is legitimate before any configuration or state data is sent to or received from the SSH client. Neither side should establish a NETCONF over SSH connection with an unknown, unexpected, or incorrect identity on the opposite side.

Configuration or state data may include sensitive information, such as usernames or security keys. So, NETCONF requires communications channels that provide strong encryption for data privacy. This document defines a NETCONF over SSH mapping that provides for support of strong encryption and authentication.

This document requires that SSH servers default to allowing access to the "netconf" SSH subsystem only when using a specific TCP port assigned by IANA for this purpose. This will allow NETCONF over SSH traffic to be easily identified and filtered by firewalls and other network nodes. However, it will also allow NETCONF over SSH traffic to be more easily identified by attackers.

This document also recommends that SSH servers be configurable to allow access to the "netconf" SSH subsystem over other ports. Use of that configuration option without corresponding changes to firewall or network device configuration may unintentionally result in the ability for nodes outside of the firewall or other administrative boundaries to gain access to the "netconf" SSH subsystem.

RFC 4742 assumes that the end-of-message (EOM) sequence,]]>]]>, cannot appear in any well-formed XML document, which turned out to be mistaken. The EOM sequence can cause operational problems and open space for attacks if sent deliberately in RPC messages. It is however believed that the associated threat is not very high. This document still uses the EOM sequence for the initial <hello> message to avoid incompatibility with existing implementations. When both peers implement base:1.1 capability, a proper framing protocol (chunked framing mechanism; see Section 4.2) is used for the rest of the NETCONF session, to avoid injection attacks.

7. IANA Considerations

Based on the previous version of this document, RFC 4742, IANA assigned the TCP port 830 as the default port for NETCONF over SSH sessions.

IANA had also assigned "netconf" as an SSH Subsystem Name, as defined in [RFC4250], as follows:

Subsystem Name	Reference
netconf	RFC 4742

IANA updated these allocations to refer to this document.

8. Acknowledgements

Ted Goddard was a co-author on earlier versions of this document.

This document was written using the xml2rfc tool described in RFC 2629 [RFC2629].

Extensive input was received from the other members of the NETCONF design team, including: Andy Bierman, Weijing Chen, Rob Enns, Wes Hardaker, David Harrington, Eliot Lear, Simon Leinen, Phil Shafer, Juergen Schoenwaelder, and Steve Waldbusser. The following people have also reviewed this document and provided valuable input: Olafur Gudmundsson, Sam Hartman, Scott Hollenbeck, Bill Sommerfeld, Balazs Lengyel, Bert Wijnen, Mehmet Ersue, Martin Bjorklund, Lada Lothka, Kent Watsen, and Tom Petch.

9. References

9.1. Normative References

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

[RFC4250] Lehtinen, S. and C. Lonvick, "The Secure Shell (SSH) Protocol Assigned Numbers", RFC 4250, January 2006.

[RFC4252] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Authentication Protocol", RFC 4252, January 2006.

[RFC4253] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, January 2006.

[RFC4254] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Connection Protocol", RFC 4254, January 2006.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

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

9.2. Informative References

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

Appendix A. Changes from RFC 4742

This section lists major changes between this document and RFC 4742.

- Introduced the new chunked framing mechanism to solve known security issues with the EOM framing.
- Extended text in Security Considerations; added text on EOM issues.
- Added examples to show new chunked encoding properly; highlighted the location of new lines.
- Added text for NETCONF username handling following the requirements on usernames in [RFC6241].
- Changed use of the terms "client/server" and "manager/agent" to "SSH client/server" and "NETCONF client/server".
- Consistently used the term "operation", instead of "command" or "message".
- Integrated errata verified for RFC 4742 as of the date of publication of this document. See errata for RFC 4742 at http://www.rfc-editor.org.

Author's Address

Margaret Wasserman
Painless Security, LLC
356 Abbott Street
North Andover, MA 01845
USA

Phone: +1 781 405-7464
EMail: mrw@painless-security.com
URI: http://www.painless-security.com