Network Working Group M. Wasserman
Internet-Draft Nokia
Expires: April 18, 2004 T. Goddard
Wind River
October 19, 2003

Using the NETCONF Configuration Protocol over Secure Shell (SSH)
draft-ietf-netconf-ssh-00.txt

Status of this Memo

Copyright Notice

Abstract

This document describes a simple method for invoking and running the NETCONF configuration protocol within a Secure Shell (SSH) session as an SSH subsystem. Some features of the NETCONF protocol are not suited for use in a single shell session, and those limitations are described here.

Table of Contents

1. Introduction

The NETCONF protocol [1] 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 XMLCONF can be used within a Secure Shell (SSH) session, using the SSH connection protocol [2] over the SSH transport protocol [3].

NETCONF is defined as a multi-channel protocol, with separate communications channels for session management, protocol operations and notifications. In this document, however, we have defined a mapping to run NETCONF over a single SSH session (a single SSH channel of type "session", see section 4 of [2]). This mapping will allow NETCONF to be executed from the a secure shell session, by a user or a simple script. Mapping NETCONF to a single SSH session does impose some limitations on the use of NETCONF over SSH. In particular, the <rpc-progress> and <rpc-abort> elements are not supported, NETCONF capabilities must be exchanged over the same channel as the NETCONF RPC commands, and NETCONF notifications, if enabled, will also be transmitted over the same channel.

Throughout this document, the terms "client" and "server" are used to refer to the two ends of the SSH transport connection. The client actively opens the SSH connection, and the server passively listens for the incoming SSH connection.

The terms "manager" and "agent" are used to refer to the two ends of the NETCONF protocol session. The manager issues NETCONF RPC commands, and the agent replies to those commands. Depending upon negotiated capabilities, the manager may also receive NETCONF notifications and the agent may send notifications.

2. Starting NETCONF over SSH

To run NETCONF over SSH, the client will first establish an SSH transport connection using the SSH transport protocol, and the client and server will exchange keys for message integrity and encryption. The client will then invoke the "ssh-userauth" service to authenticate the user, as described in the SSH authentication protocol [4]. Once the user has been successfully authenticated, the client will invoke the "ssh-connection" service, also known as the SSH connection protocol.

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

Once the SSH session has been established, the user (or script) will invoke NETCONF as an SSH subsystem called "netconf". 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 printed at shell start-up.

To the user (or script), running NETCONF as an SSH subsystem may look similar to the following example. Although this example shows the text transmitted by both sides, the server MUST NOT echo the commands that it receives back to the client.

```
   <!-- The user (or script) invokes the SSH subsystem.  Depending upon
   the configuration of the client and server, the passphrase prompt
   may not be issued or may be replaced by a password prompt. -->

   [user@client]$ ssh -s server.example.org netconf
   Enter passphrase for key '/foo/.ssh/id_dsa':

   <!-- The NETCONF subsystem running on the server sends a complete
   XML document to the client. -->

   <?xml version="1.0" encoding="UTF-8"?>
   <hello>
       <capabilities>
           <capability>http://ietf.org/xmlconf/1.0/base</capability>
           <capability>http://ietf.org/xmlconf/1.0/agent</capbability>
           <capability>http://ietf.org/xmlconf/1.0/base#lock</capability>
       </capabilities>
   </hello>

   <!-- The client sends a complete XML document to the server. -->
```

```
<?xml version="1.0" encoding="UTF-8"?>
<hello>
    <capabilities>
        <capability>http://ietf.org/xmlconf/1.0/base</capability>
        <capability>http://ietf.org/xmlcong/1.0/manager</capability>
        <capability>http://ietf.org/xmlconf/1.0/base#lock</capability>
    </capabilities>
</hello>
```

While the NETCONF subsystem is active, the NETCONF manager can interact with the NETCONF agent by sending complete XML documents containing NETCONF RPC elements, and the NETCONF server will respond by sending complete XML documents containing appropriate RPC replies.

2.1 Capabilities Exchange

As indicated in the example above, the server MUST indicate its capabilities by sending an XML document containing a <hello> element as soon as the NETCONF session is established. The user (or the user's expect script) can parse this message to determine which NETCONF capabilities are supported by the server.

The client must also send an XML document containing a <hello> element to indicate the client's capabilities to the server. The document containing the <hello> element must be the first XML document that the client sends after the NETCONF session is

established.

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

2.2 Reversability of Connections

The NETCONF protocol is reversible -- either the manager or the agent may initiate the session-layer or transport connection. Once the session is established, the NETCONF capabilities exchange will be used to indicate which side of the connection is the agent and which is the manager, as indicated in the previous example. If there is no agreement, each side MUST close the transport connection and log an error.

In order to provide for reversability when used over SSH, it may be necessary for either the NETCONF agent or the NETCONF manager to have a well known host key, as it is always required for the SSH server to have a well known host key. Thus, the server will authenticate itself to the client with its host key. The client will then authenticate itself with any allowable mechanism, as specified in [4]. The authenticated principle is then passed to NETCONF.

Because the use of NETCONF may involve transferring sensitive configuration information in either direction, both the client and server must be authenticated and all of the data exchanged must be encrypted. If either the client or server fails to successfully authenticate itself, or if it is not possible to establish an encrypted session, the NETCONF session MUST be aborted.

3. Using NETCONF over SSH

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

<rpc-reply> elements.

To continue the example given above, an XMLCONF over SSH session to retrieve a set of configuration information might look like this:

```
<!-- The manager sends an XML document containing an <rpc>
element. -->

<?xml version="1.0" encoding="UTF-8"?>
<rpc id="101" xmlns="http://ietf.org/netconf/1.0/base">
    <get-config>
        <source> <running/> </source>
        <config xmlns="http://example.com/schema/1.2/config">
            <users/>
        </config>
        <format>xml</format>
    </get-config>
</rpc>

<!-- The agent responds with an XML document containing an
<rpc-reply> element. -->

<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply id="101" xmlns="http://ietf.org/netconf/1.0/base">
    <config xmlns="http://example.com/schema/1.2/config">
        <users>
            <user><name>root</name><type>superuser</type></user>
            <user><name>fred</name><type>admin</type></user>
            <user><name>barney</name><type>admin</type></user>
        </users>
    </config>
</rpc-reply>
```

There are two NETCONF protocol operations that are not supported when running NETCONF over SSH, the <rpc-progress> and <rpc-abort> operations. These operations use the NETCONF management channel to allow the processing of out-of-band operations that affect RPC processing on the operations channel. Since this document defines a single-channel mechanism for using NETCONF over SSH, these operations cannot be supported in this transport mapping.

In this mapping, there is no way to obtain a progress indication regarding an outstanding RPC request, and the only way to abort an RPC request before it completes is to terminate the SSH session.

4. Sending NETCONF Notifications over SSH

The SSH protocol has the capability to support multiple sessions, and therefore, theoretically to support multiple NETCONF channels. However, because the NETCONF over SSH mapping is designed for simplified scripting, use of this mapping for such multiple purposes is not supported.

Instead, if both the manager and agent indicate support for the notification capability and the manager issues a <notification-open> RPC command, notifications may be sent over the SSH session, interleaved with NETCONF RPC commands and responses. Notifications are transmitted and received as described in RFC 3195 [5], with the exception that authentication information is passed from the SSH layer instead of the BEEP layer.

Once the client or the server begins sending an XML document, it must suspend all other output (i.e. other XML documents) until the document has been sent in its entirety. This means that asynchronous notifications may be delayed while waiting for the transmission of other documents to be completed. It is recommended that if an agent has at least one notification pending and at least one response pending that the notification(s) be sent first. If a manager deems that notifications are particularly time-sensitive, it may open another NETCONF session that is used only for notifications.

It is acceptable for a NETCONF manager to be sending a command to the agent, while the agent is simultaneously sending a response or notification to the manager.

Once the manager has requested that the agent send notifications, via a <notification-open> RPC message, the agent may send notification until the SSH session is closed.

5. Exiting the NETCONF Subsystem

Exiting NETCONF is accomplished using the <kill-session> operation. When a <kill-session> command is issued by the manager, the agent shall respond, terminate the SSH session, and close the TCP connection.

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

```
<!-- The manager sends an XML document containing a <kill-session>
operation. Question: Where do we get the session-id? Should it be
sent in the <hello> message? -->

<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="102" xmlns="http://ietf.org/xmlconf/1.0/base">
     <kill-session>
         <session-id>0</session-id>
     </kill-session>
</rpc>

<!-- The agent returns an "OK" reply. -->

<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply id="102" xmlns="http://ietf.org/netconf/1.0/base">
    <ok/>
</rpc-reply>

<!-- The NETCONF subsystem exits, ending the SSH session and returning
the user (or script) to the local shell prompt. -->

[user@client]$
```

6. Running NETCONF from an SSH Shell

The techniques described in this document could be used to access the NETCONF protocol over the SSH shell session, or from other shell types such as a console session or a Telnet [7] connection. However, there are serious security implications associated with allowing NETCONF access via any method that does not provide strong support for user authentication, server authentication and data privacy. See the Security Considerations section for more details.

If the server supports NETCONF invocation from an SSH shell session, the user may choose to invoke a NETCONF program from the shell command line. This would involve using SSH to establish a shell session, and entering the name of a NETCONF program (with the full path, if necessary) at the remote shell prompt.

6.1 Starting a NETCONF Shell Session

To the user, the establishment of an SSH shell and the invocation of the NETCONF program may look similar to the following example:

```
   <!-- The user enters an SSH shell session. -->

   [user@client]$ ssh server.example.org
   user@server.example.org's password: ********

   <!-- At the shell prompt, the user invokes the NETCONF program, which
   in this example is called 'netconf', but which might have different
```

```
names on different systems. -->

[user@server]$ netconf

<!-- The NETCONF program sends an XML document to the client. -->

<?xml version="1.0" encoding="UTF-8"?>
<hello>
    <capabilities>
        <capability>http://ietf.org/xmlconf/1.0/base</capability>
        <capability>http://ietf.org/xmlconf/1.0/base#lock</capability>
    </capabilities>
</hello>
```

6.2 Exiting a NETCONF Shell Session

When the user has run NETCONF from a shell, he will need to exit the NETCONF program using the <kill-session> operation, and then exit the remote shell to return to the local shell. To continue the example used in previous sections, an existing NETCONF shell session could be closed as follows:

```
<!-- The manager sends an XML document containing an <kill-session>
operation Question: Where do we get the session-id?  Should it be
sent in the <hello> message? -->

<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="102" xmlns="http://ietf.org/xmlconf/1.0/base">
     <kill-session>
         <session-id>0</session-id>
     </kill-session>
</rpc>

<!-- The agent returns an "OK" reply. -->

<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply id="102" xmlns="http://ietf.org/netconf/1.0/base">
    <ok/>
```

```
   </rpc-reply>

   <!-- The NETCONF program exits, returning the user to the SSH prompt.
   The user then types 'exit' to exit the SSH shell and return to the
   local shell. -->

   [user@server]$ exit
   [user@client]$
```

7. Security Considerations

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

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

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

If the NETCONF server provides remote shell access through insecure protocols, such as Telnet, care should be taken to prevent execution of the NETCONF program when strong user authentication or data privacy is not available. Because it may be difficult or impossible, in some operating environments, to determine whether a shell command was accessed over a secure protocol, such as SSH, or an insecure protocol, such as Telnet, it may be necessary to disable insecure shell access to the system to prevent insecure access to a NETCONF program. Alternatively, it would be possible to disable NETCONF access from the command line, only allowing NETCONF to be accessed through invocation of the SSH 'netconf' subsystem.

Because NETCONF data is being sent over the standard SSH protocol port mapping (to the NETCONF subsystem or shell), use of this protocol mapping would allow NETCONF data to be transferred over a firewall boundary that has open access for SSH connections. Network policies which restrict NETCONF operations to within a trusted network may clash with other policies that allows network-external SSH access to internal ssh services.

8. Acknowledgements

This document was written using the xml2rfc tool described in RFC 2629 [8].

Extensive input was received from the members of the NETCONF design team, including: Andy Bierman, Weijing Chen, Rob Enns, Wes Hardaker, Eliot Lear, Simon Leinen, Phil Shafer, Juergen Shoenwaelder and Steve

Waldbusser. The following people have also reviewed this document and provided helpful input: Bill Sommerfeld.

Normative References

[1] Enns, R., "NETCONF Configuration Protocol", draft-ietf-netconf-prot-00 (work in progress), August 2003.

[2] Ylonen, T., Kivinen, T., Rinne, T. and S. Lehtinen, "SSH Connection Protocol", draft-ietf-secsh-connect-17 (work in progress), July 2003.

[3] Ylonen, T., Kivinen, T., Saarinen, M., Rinne, T. and S. Lehtinen, "SSH Transport Layer Protocol", draft-ietf-secsh-transport-16 (work in progress), July 2003.

[4] Ylonen, T., Kivinen, T., Saarinen, M., Rinne, T. and S. Lehtinen, "SSH Authentication Protocol", draft-ietf-secsh-userauth-17 (work in progress), July 2003.

[5] New, D. and M. Rose, "Reliable Delivery for syslog", RFC 3195, November 2001.

Informative References

[6] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

[7] Postel, J. and J. Reynolds, "Telnet Protocol Specification", STD 8, RFC 854, May 1983.

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

Authors' Addresses

Margaret Wasserman
Nokia
5 Wayside Road
Burlington, MA 01803
USA

Phone: +1 781 993 3858
EMail: margaret.wasserman@nokia.com
URI: http://www.nokia.com

Ted Goddard
Wind River
#180, 6815-8th St. N.E.
Calgary, AB T2E 7H7
Canada

Phone: +1 403 730 7590
EMail: ted.goddard@windriver.com

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

Full Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.