Using the NETCONF Configuration Protocol over Secure Shell (SSH)
draft-ietf-netconf-ssh-06.txt

Status of this Memo

   By submitting this Internet-Draft, each author represents that any
   applicable patent or other IPR claims of which he or she is aware
   have been or will be disclosed, and any of which he or she becomes
   aware will be disclosed, in accordance with Section 6 of BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF), its areas, and its working groups. Note that
   other groups may also distribute working documents as Internet-
   Drafts.

   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."

   The list of current Internet-Drafts can be accessed at
   http://www.ietf.org/ietf/1id-abstracts.txt.

   The list of Internet-Draft Shadow Directories can be accessed at
   http://www.ietf.org/shadow.html.

   This Internet-Draft will expire on September 7, 2006.

Copyright Notice

Abstract

   This document describes a method for invoking and running the NETCONF
   protocol within a Secure Shell (SSH) session as an SSH subsystem.

Table of Contents

1. Introduction

The NETCONF protocol [I-D.ietf-netconf-prot] 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.

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. When NETCONF is run over SSH using the mapping defined in
this document, the client is always the manager, and the server is
always the agent.

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 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 [RFC4252]. 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 application)
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.
However, if a subsystem cannot be used, it should be possible for a
client to skip over any system messages that are sent at shell
start-up by searching for a NETCONF <hello> element. Note that this
may not avoid problems if system messages are recieved later in the
session.

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
<TBD>. 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 <TBD> netconf

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

3.1. Capabilities Exchange

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 application) 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.

The following example shows a capability exchange. Messages sent by
the client are marked with "C:" and message sent by the server are
marked with "S:".

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

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

As the previous example illustrates, a special character sequence,
]]>]]>, MUST be sent by both the client and the server after each XML
document in the NETCONF exchange. This character sequence cannot
legally appear in an XML document, so it can be unambigiously used to
indentify the end of the current document, allowing resynchronization
of the NETCONF exchange in the event of an XML syntax or parsing
error.

[4](#). 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 NETCONF over SSH session 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.
An agent will process RPC messages from the manager in the order in
which the are received. When the agent processes a <close-session>
command, the agent shall respond and terminate the SSH session. The
agent MUST NOT process any RPC commands received on the current
session after the <close-session> command.

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

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

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

[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 agent's configuration and state or to modify the agent's
configuration.

The identity of the server MUST be verified and authenticated by the
client according to local policy before password-based authentication
data or any configuration or state data is sent to or received from
the server. The identity of the client MUST also be verified and
authenticated by the server according to local policy to ensure that
the incoming client request is legitimate before any configuration or
state data is sent to or received from the 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 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.

This document requires that 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 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
boundary to gain access to "netconf" SSH subsystem.

7. IANA Considerations

IANA is requested to assign a TCP port number which will be the
default port for NETCONF over SSH sessions as defined in this
document.

IANA has assigned port <TBD> for this purpose.

[RFC Editor: Please replace two instances of <TBD> in Section 3 and
one instance of <TBD> in this section to the port number assigned by
IANA.]

IANA is also requested to assign "netconf" as an SSH Service Name as
defined in [RFC4250], as follows:

        Service Name   Reference
        -------------   ---------
        netconf   [This Document]

8. Acknowledgements

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, Bert
Wijnen.

9. References

9.1. Normative References

[I-D.ietf-netconf-prot]
            Enns, R., "NETCONF Configuration Protocol",
            draft-ietf-netconf-prot-11 (work in progress),
            February 2006.

[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.

9.2. Informative References

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

Authors' Addresses

   Margaret Wasserman
   ThingMagic
   One Broadway, 5th Floor
   Cambridge, MA 02142
   USA

   Phone: +1 781 405-7464
   Email: margaret@thingmagic.com
   URI: http://www.thingmagic.com

   Ted Goddard
   ICEsoft Technologies, Inc.
   Suite 300, 1717 10th St. NW
   Calgary, AB T2M 4S2
   Canada

   Phone: +1 403 663-3322
   Email: ted.goddard@icesoft.com
   URI: http://www.icesoft.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment