Secure Shell Working Group
Internet-Draft
Expires: December 10, 2011

K. Watsen
Juniper Networks

June 8, 2011

Reverse Secure Shell (Reverse SSH)
draft-kwatsen-reverse-ssh-01

Abstract

This memo presents a technique for a SSH (Secure Shell) server to initiate the underlying TCP connection to the SSH client. This role reversal is necessary in cases where the SSH client would otherwise be unable to initiate an SSH connection to the SSH server, such as a device "calling home" on its first boot.

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 December 10, 2011.

1. 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].

2. Introduction

This memo presents a technique for a SSH (Secure Shell) [RFC4251] server to initiate the underlying TCP connection to the SSH client. This role reversal is necessary in cases where the SSH client would otherwise be unable to initiate an SSH connection to the SSH server, such as a device "calling home" on its first boot.

This document uses the terms "Reverse SSH client" and "Reverse SSH server" in order to reflect the role of each peer. The Reverse SSH client is the peer that initiates the TCP connection and then starts the SSH server protocol. The Reverse SSH server is the peer that listens for and accepts TCP connections and then starts the SSH client protocol.

This RFC modifies the SSH protocol in two ways:

- Removes the restriction that the SSH Client must initiate the TCP connection.
- Defines the "hmac-*" family of public key algorithms.

This RFC additionally defines a YANG [RFC6020] module for the configuration of the Reverse SSH client running on a device.

3. Benefits to Device Management

The SSH protocol is nearly ubiquitous for device management, as it is the transport for the command-line applications `ssh`, `scp`, and `sftp` and the required transport for the NETCONF protocol [RFC4741]. However, in all these cases, the device expects to be the SSH server so that it can authenticate the user, apply security credentials, enable SSH channels to be opened, and so on. Reverse SSH allows the device to always be the SSH server regardless of which peer initiates the underlying TCP connection.

Reverse SSH is useful for both initial deployment and on-going device management. Use of Reverse SSH for initial deployment is independent of its use for on-going management.

For initial deployment, Reverse SSH may be used as a "call home" mechanism, similar to that provided by Broadband Forum TR-069 [TR069], but with the benefit of not being bound to any particular protocol (SOAP over HTTP).

For on-going management, Reverse SSH may be used to enable any of the following scenarios:

- The device may be deployed behind a NAT-ing device that doesn't provision an external address and port to connect to.
- The device may be deployed behind a firewall that doesn't allow SSH access to the internal network.
- The device may be configured in "stealth mode" with no open ports
- The device may access the network in a way that dynamically assigns it an IP address and is not configured to use a service to register its dynamically-assigned IP address to a well-known domain name.
- The operator prefers to have one open-port to secure in the data center, rather than have an open port on each device in the network.

One key benefit of using SSH as the transport protocol for Reverse SSH is its ability to multiplex an unspecified number of independently flow-controlled TCP sessions on top of a single encrypted tunnel [RFC4254]. This feature is valuable as the device only needs to be configured to initiate a single Reverse SSH connection regardless the number the TCP-based protocols the application wishes to support. For instance, the application may "pin up" a channel for each distinct type of asynchronous notification the device supports (logs, traps, backups, etc.) and dynamically open/close channels as needed by its runtime. Lastly, using SSH channels has been found to be more straightforward and supported than using other multiplexing protocols such as Block Extensible Exchange Protocol (BEEP) [RFC3080].

4. Protocol Overview

The Reverse SSH client's perspective

- The Reverse SSH client initiates a TCP connection to the Reverse SSH server on the IANA-assigned SSH port (port 22)

- Immediately after the TCP session starts, the Reverse SSH client starts the SSH server protocol using the accepted TCP connection. That is, the Reverse SSH client sends it's SSH host key during the SSH key exchange.

The Reverse SSH server's perspective

- The Reverse SSH server listens for TCP connections on the IANA-assigned SSH port (port 22)

- The Reverse SSH server accepts an incoming TCP connection and immeditately starts the SSH client protocol. That is, the Reverse SSH server will need to authenticate its peer's SSH host key during the SSH key exchange.

Note: in order to enable the Reverse SSH server to identify the Reverse SSH client and automatically authenticate its SSH host key, each peer SHOULD only advertise support for one of the following host key algorithms:

Algorithm	Reference
x509v3-ssh-dss	[RFC6187]
x509v3-ssh-rsa	[RFC6187]
x509v3-rsa2048-sha256	[RFC6187]
x509v3-ecdsa-sha2-*	[RFC6187]
hmac-ssh-dss	[RFCXXXX]
hmac-ssh-rsa	[RFCXXXX]
hmac-rsa2048-sha256	[RFCXXXX]
hmac-ecdsa-sha2-*	[RFCXXXX]

5. The hmac-* Public Key Algorithms

This section defines a family of public host key algorithms that can be used to both identify the SSH server and enable its host key to be automaticaly authenticated.

The algorithms presented in this section rely on a symmetric HMAC key to convey trust. This is in contrast to the PKI based authentication model used by the x.509 based public key algorithms [[RFC6187]]. Using an HMAC key, which can be interactively provided to the SSH Server, enables Reverse SSH to be used in deployments where it's not possible for a x.509 Certificate Authority to sign the device's certificate in time. For instance, when the device is "calling home" the first time in order to receive its full configuration.

The HMAC-based host keys defined in this specification mirror those defined in [RFC6187]. These host-keys are to be treated the same way as in [RFC6187], except that the the peer authenticates the host key via an HMAC, instead of PKIX.

Regardless of which underlying host key is used, the format of the hmac-* based public key is as follows:

```
string server-id
string host-key
string hmac
```

The "server-id" field encodes a user-configured unique identifier for the SSH Server. This field is necessary as the Reverse SSH client MAY not be identifiable from its TCP session's source address. For instance, the Reverse SSH client may be "calling home" for the first time or have a dynamically assigned address (DHCP, NAT, etc.).

The "host-key" field is the SSH Server's corresponding SSH host key. For instance, if the "hmac-ssh-rsa" public key was negotiated during key exchange, this field would encode the "ssh-rsa" host key.

The "hmac" field is the value produced using the MAC algorithm negotiated during key exchange over the selected host key and a user-configured HMAC key [[RFC2104]]

6. Device Configuration

For devices supporting NETCONF [RFC4741], this section defines a YANG [RFC6020] module to configure the Reverse SSH client on the device. For devices that do not support NETCONF, this section illustrates what its configuration data model SHOULD include.

This YANG module enables a NETCONF client to generically manage the NETCONF server's Reverse SSH client configuration without needing to understand a device-specific data-model. This is important as a normalized configuration is necessary to bootstrap multi-vendor devices for their "initial deployment". The definition of a YANG module also ensures that key features are enabled such as supporting more than one application, more than one server per application, and the definition of a reconnection strategy.

This RFC does not attempt to define any strategy for how an initial deployment might obtain its bootstrapping "call home" configuration (address to connect to, signature algorithm to use, authentication credentials to use, etc.). That said, implementations may consider use of a DHCP server or a USB pen drive as viable options for these kinds of deployments.

Configuration Example

```
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <reverse-ssh xmlns="urn:ietf:params:xml:ns:yang:ietf-reverse-ssh">
      <applications>
         <application>
            <name>config-mgr</name>
            <description>
               This entry requests the device to periodically
               connect to the Configuration Manager application
            </description>
            <device-id>9876436534</device-id>
            <periodic-connection>
               <interval-mins>5</interval-mins>
               <linger-secs>20</linger-secs>
            </periodic-connection>
            <symmetric-authentication>
               <algorithm>hmac-sha1</algorithm>
               <hmac-key>secret</hmac-key>
            </symmetric-authentication>
            <servers>
               <server>
                  <host>config-mgr1.acme.com</host>
                  <port>7022</port>
               </server>
               <server>
                  <host>config-mgr2.acme.com</host>
                  <port>7022</port>
               </server>
             </servers>
             <keep-alive-strategy>
                <interval-secs>5</interval-secs>
```

```
                  <count-max>3</count-max>
               </keep-alive-strategy>
               <reconnect-strategy>
                  <start-with>last-connected</start-with>
                  <interval-secs>10</interval-secs>
                  <count-max>4</count-max>
               </reconnect-strategy>
            </application>
            <application>
               <name>log-monitor</name>
               <description>
                  This entry requests the device to mantain a
                  persistent connection to the Log Monitoring
                  application
               </description>
               <device-id>device-23.53432</device-id>
               <persistent-connection/>
               <assymmetric-authentication>
                  <algorithm>rsa-sha1</algorithm>
                  <assymetric-key>secret</assymetric-key>
               </assymmetric-authentication>
               <servers>
                  <server>
                     <host>log-mon1.acme.com</host>
                     <port>7514</port>
                  </server>
                  <server>
                     <host>log-monitor2.acme.com</host>
                     <port>7514</port>
                  </server>
               </servers>
               <keep-alive-strategy>
                  <interval-secs>5</interval-secs>
                  <count-max>3</count-max>
               </keep-alive-strategy>
               <reconnect-strategy>
                  <start-with>last-connected</start-with>
                  <interval-secs>10</interval-secs>
                  <count-max>4</count-max>
               </reconnect-strategy>
            </application>

         </applications>
      </reverse-ssh>
   </config>
```

The YANG Module

```
module ietf-reverse-ssh {

   namespace "urn:ietf:params:xml:ns:yang:ietf-reverse-ssh";

   prefix "rssh";

   import ietf-inet-types { prefix inet; }

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

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

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

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

      Editor: Kent Watsen
              <mailto:kwatsen@juniper.net>";


   revision 2011-04-26 {
       description "Initial conception";
       reference "RFC XXXX: Reverse SSH";
   }
   // RFC Ed.: replace XXXX with actual
   // RFC number and remove this note


   container reverse-ssh {
       container applications {
           description
               "All the application that the device
                initiates Reverse SSH connections to";
           list application {
               key name;
               min-elements 1;
               leaf name {
                   mandatory true;
                   type string {
                       length 1..32;
                   }
```

```
                    description
                       "The name of the specific application";
                }
                leaf description {
                    type string;
                    description
                      "An optional description for the application";
                }
                leaf device-id {
                    type string {
                        length 1..32;
                    }
                    description
                       "The identifier the device uses to
                        identify itself to this application.  If
                        not specified, the device will use it's
                        serial-number (not recommneded)";
                }
                choice connection-type {
                    description "Indicates the application's
                                 preference for how the device's
                                 connection is maintained.";
                    default persistent-connection;
                    leaf persistent-connection {
                        type empty;
                    }
                    container periodic-connection {
                        leaf interval-mins {
                            type uint8;
                            default 5;
                            units minutes;
                            description
                               "The amount of unconnected time the
                                device will wait until establishing
                                a connection just in case the
                                application has some data pending
                                to send it.  The device MAY
                                establish a connection before this
                                time if it has data is needs to
                                send to the device.";
                        }
                        leaf linger-secs {
                            type uint8;
                            default 30;
                            units seconds;
                            description
                               "The amount of time it should wait
                                after last receiving from or sending
```

```
                    data to the device before closing
                    the connection.  This optimization
                    trades off the latency for
                    resources.";
                }
            }
        }
        choice authentication-strategy {
            mandatory true;
            container symmetric-authentication {
                leaf algorithm {
                    default hmac-sha1;
                    type enumeration {
                        enum hmac-md5;
                        enum hmac-sha1;
                        enum hmac-sha256;
                    }
                }
                leaf hmac-key {
                    mandatory true;
                    type string;  // secret
                }
            }
            container assymmetric-authentication {
                leaf algorithm {
                    default rsa-sha1;
                    type enumeration {
                        enum rsa-sha1;
                    }
                }
                leaf assymetric-key {
                    mandatory true;
                    type string;  // secret
                }
            }
        }
        container servers {
            description
                "An ordered listing of the application's
                 servers that the device should attempt
                 connecting to.";
            list server {
                key host;
                min-elements 1;
                ordered-by user;
                leaf host {
                    mandatory true;
                    type inet:host;
```

```
                    description
                        "IP address or domain-name for
                         the server";
                }
                leaf port {
                    type inet:port-number;
                    description
                        "The IP port for this server.
                         The device will use the
                         IANA-assigned port if not
                         specified.";
                }
            }
        }
        container keep-alive-strategy {
            leaf interval-secs {
                type uint8;
                units seconds;
                default 15;
                description
                  "Sets a timeout interval in seconds after
                   which if no data has been received from
                   the client, a message will be sent to
                   request a response from the SSH client.
                   A value of '0' indicates that no messages
                   should be sent.";
            }
            leaf count-max {
                type uint8;
                default 3;
                description
                  "Sets the number of keep alive messages
                   that may be sent without receiving any
                   response from the SSH client before
                   assuming the SSH client is no longer
                   alive.  If this threshold is reached
                   the device will disconnect the SSH
                   session.  The keep alive interval timer
                   is reset after each transmission.  Thus,
                   an unresponsive SSH client will be
                   disconnected after approximately
                   'count-max * interval-secs' seconds.";
            }
        }
        container reconnect-strategy {
            leaf start-with {
                default first-listed;
                type enumeration {
```

```
                            enum first-listed;
                            enum last-connected;
                        }
                    }
                    leaf interval-secs {
                        type uint8;
                        units seconds;
                        default 5;
                        description
                          "time delay between connection attempts";
                    }
                    leaf count-max {
                        type uint8;
                        default 3;
                        description
                          "num times try to connect to a server";
                    }
                }
            }
        }

    }
}
```

7. Security Considerations

This RFC deviates from standard SSH protocol usage by allowing the SSH server to initiate the TCP connection. This conflicts with section 4 of the SSH Transport Layer Protocol RFC [RFC4253], which states "The client initiates the connection". This role reversal, however, does not alter the fundamentals for how SSH client and SSH server authenticate eachother, and thus doesn't affect the security of the solution.

This RFC defines new HMAC-based public key algorithms. Implementations SHOULD use a MAC algorithm and an HMAC-key such that the cryptographic strength of the HMAC is not less than the strength of the host key it vouches for.

The HMAC-based public key algorithms specify a "server-id" field that is passed in the clear. The server-id field SHOULD NOT contain a value that might provide an observer any undue information about the device. Specifically, it is NOT RECOMMENDED to use the device's serial number for its "server-id", as it may reveal the device's model-number and/or manufacturing date.

The hmac-* public key algorithms require the application consume the

server-id field without being able to first verify that it is the value the device sent. The application must use the server-id value to lookup the device's record in a local datastore in order to obtain the HMAC-key needed to authenticate the HMAC. The application must be sure to process the server-id carefully as it may have been purposely encoded to illicit unexpected behaviour.

An attacker could DoS the application using valid "server-id" values, forcing the application to perform computationally expensive operations, only to deduce that the attacker doesn't posses a valid key. This is no different than any secured service and all common precautions apply (e.g. blacklisting the source address after a set number of unsuccessful login attempts).

8. IANA Considerations

Consistent with Section 8 of [[RFC4251]] and Section 4.6 of [[RFC4250]], this document makes the following registrations:

In the Public Key Algorithm Names registry:

- The SSH public key algorithm "hmac-ssh-dss".
- The SSH public key algorithm "hmac-ssh-rsa".
- The SSH public key algorithm "hmac-rsa2048-sha256".
- The family of SSH public key algorithm names beginning with "hmac-ecdsa-sha2-" and not containing the at-sign ('@').

9. References

9.1. Normative References

[RFC2104] Krawczyk, H., Bellare, M., and R. Centti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

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

[RFC3080] Rose, M., Ed., "The Blocks Extensible Exchange Protocol Core", RFC 3080, March 2001.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications

Version 2.1", RFC 3447, February 2003.

[RFC4231] Nystrom, M., "Identifiers and Test Vectors for HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512", RFC 4231, December 2005.

[RFC4250] Lehtinen, S. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Assigned Numbers", RFC 4250, December 2005.

[RFC4251] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Architecture", RFC 4251, January 2006.

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

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

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

[RFC4741] Enns, R., Ed., "NETCONF Configuration Protocol", RFC 4741, December 2006.

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

[RFC6125] Saint-Andre, PSA. 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.

[RFC6187] Igoe, K. and D. Stebila, "X.509v3 Certificates for Secure Shell Authentication", RFC 6187, March 2011.

9.2. Informative References

[TR069] The Broadband Forum, "TR-069 Amendemnt 3, CPE WAN Management Protocol", November 2010.

Author's Address

Kent Watsen
Juniper Networks

Email: kwatsen@juniper.net