Network Working Group
Internet-Draft
Intended status: Informational
Expires: December 12, 2013

M. Joseph
J. Susoy
P6R, Inc
June 14, 2013

P6R's Secure Shell Public Key Subsystem
draft-joseph-pkix-p6rsshextension-02.txt

Abstract

The Secure Shell Public Key Subsystem protocol defines a key distribution protocol to provision an SSH server with user's public keys. However, that protocol is limited to provisioning an SSH server. This document describes a new protocol that builds on the protocol defined in RFC 4819 to allow the provisioning of keys and certificates to a server using the SSH transport.

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 12, 2013.

Copyright Notice

The new protocol allows the calling client to organize keys and certificates in different namespaces on a server. These namespaces can be used by the server to allow a client to configure any application running on the server (e.g., SSH, KMIP, SNMP).

The new protocol provides a server-independent mechanism for clients to add public keys, remove public keys, add certificates, remove certificates, and list the current set of keys and certificates known by the server by namespace (e.g., list all public keys in the SSH namespace).

Rights to manage keys and certificates in a specific namespace are specific and limited to the authorized user and are defined as part of the server's implementation. The described protocol is backward compatible to version 2 defined by RFC 4819.

Table of Contents

1. Introduction

This document describes a new protocol based on the protocol defined in RFC 4819 that can be used to configure public keys and certificates in an implementation-independent fashion. The addition of the concept of a namespace which allows the client to organize keys and certificates by application or organizational structure is added to the protocol's operations.

P6R's Secure Shell Public Key Subsystem has been designed to run on top of the Secure Shell transport layer [3] and user authentication protocols [4]. It provides a simple mechanism for the client to manage public keys and certificates on the server. Uploaded keys and certificates are meant to be able to configure all protocols running on a server (e.g., SSH, SSL, KMIP [6]) that use keys and certificats as well as applications that run on a server.

This document should be read only after reading the Secure Shell Public Key Subsystem [1] document. The new protocol described in this document builds on and is meant to be backwards compatible with the protocol described in [1].

2. 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 [2].

3. Overview of extensions to the Public Key Subsystem

The Public Key Subsystem provides a server-independent mechanism for clients to add public keys, remove public keys, list the current public keys known by the server, add certificates, remove certificates, and list the current set of certificates known by the server. This secure key distribution mechanism is implemented by a new SSH subsystem with the name of "publickey@p6r.com".

3.1. Extended Status Codes

The status code gives the status in a more machine-readable format (suitable for localization), and can have the following values:

```
SSH_PUBLICKEY_CERTIFCATE_NOT_FOUND         192
SSH_PUBLICKEY_CERTIFCATE_NOT_SUPPORTED     193
SSH_PUBLICKEY_CERTIFCATE_ALREADY_PRESENT   194
SSH_PUBLICKEY_ACTION_NOT_AUTHORIZED        195
SSH_PUBLICKEY_CANNOT_CREATE_NAMESPACE      196
```

The meaning of the failure codes is as implied by their names. See Security Considerations for the use of the failure code:

SSH_PUBLICKEY_ACTION_NOT_AUTHORIZED.

3.2. The Version Packet

Both sides MUST start a connection by sending a version packet that indicates the version of the protocol they are using.

```
string "version"
uint32 protocol-version-number
```

This document defines version 3 of the new protocol. We are using version 3 so that it can be backward compatible with the protocol defined by RFC 4819 [1].

3.3. The Namespace Attribute

The "namespace" attribute is added as an extension to what was described in RFC 4819. The purpose of this attribute is to be able to organize the uploaded keys and certificates into groups where each group represents an application or organization structure. This attribute is a string that should not be longer than 300 characters and MUST be specified in UTF-8 format [5].

This new protocol uses the "ssh" namespace for the manipulation of public keys in an SSH server and should be considered as the default namespace when none is provided.

As a convention, namespaces used for protocols are lower case strings of the protocol's standard abbreviation. For example, "ssl" should be the namespace used for the Secure Sockets Layer protocol. Namespaces for applications should contain the product and vendor's name. To help determine what namespaces already exist on a server a new operation "listnamespaces" is defined in Section 4.

4. New Operations

P6R's Public Key Subsystem extendeds the functionality defined in RFC 4819 with the following operations: add-certificate, remove-certificate, list-certificates, and listnamespaces.

4.1. Adding a Certificate

If the client wishes to add a certificate, the client sends:

```
string    "add-certificate"
string    certificate format name
string    certificate blob
boolean   overwrite
uint32    attribute-count
 string    attrib-name
 string    attrib-value
 bool      critical
```

repeated attribute-count times

This request MUST include at least the "namespace" attribute so that the server knows where to save the certificate to. Only one namespace attribute can be used per add-certificate request. It is possible for the same user to save the same certificate into multiple namespaces but this must be done with several separate add-certificate requests.

If the namespace appearing in an add-certificate request does not already exist on a server then it is created by this operation. However, if the user is not authorized to create a namespace the server MUST return SSH_PUBLICKEY_CANNOT_CREATE_NAMESPACE.

If the overwrite field is false and the specified certificate already exists in the given namespace, the server MUST return SSH_PUBLICKEY_CERTIFICATE_ALREADY_PRESENT. If the server returns this, the client SHOULD provide an option to the user to overwrite the certificate. If the overwrite field is true and the specified key already exists in the given namespace, but cannot be overwritten, the server MUST return SSH_PUBLICKEY_ACCESS_DENIED.

However, a user may not be authorized to add a certificate to the specified namespace. If the user does not have permission to add a certificate then the server MUST return SSH_PUBLICKEY_ACTION_NOT_AUTHORIZED.

Examples of possible "certificate format name" are: "X509", "PGP".

The format of public key and certificate blobs are detailed in Section 6.6, "Public Key Algorithms" of the SSH Transport Protocol document [3].

4.2. Removing a Certificate

If the client wishes to remove a certificate, the client sends:

```
string    "remove-certificate"
string    certificate format name
string    certificate blob
uint32    attribute-count
 string    attrib-name
 string    attrib-value
repeated attribute-count times
```

This request MUST include at least the "namespace" attribute so that the server knows where to delete the certificate from. Only one namespace attribute can be used per remove-certificate request. The server MUST attempt to remove the certificate from the the appropriate location.

However, a user may not be authorized to remove a certificate from the specified namespace. If the user does not have permission to remove the certificate then the server MUST return SSH_PUBLICKEY_ACTION_NOT_AUTHORIZED.

Examples of possible "certificate format name" are: "X509", "PGP".

4.3. Listing Certificates

If the client wishes to list the known certificates, the client sends:

```
string    "list-certificates"
```

The server will respond with zero or more of the following responses:

```
string    "certificate"
string    certificate format name
string    certificate blob
uint32    attribute-count
 string    attrib-name
 string    attrib-value
repeated attribute-count times
```

There is no requirement that the responses be in any particular order. Whilst some server implementations may send the responses in some order, client implementations should not rely on responses being in any order.

This response MUST include at least the "namespace" attribute so that a client can tell which namespace the certificate resides.
Only one namespace attribute can be used per list-certificate request.

Following the last "certificate" response, a status packet MUST be sent.

4.4. Listing Namespaces

If the client wishes to know existing namespaces on the server, it sends:

```
string    "listnamespaces"
```

The server will respond with zero or more of the following responses:

```
string    "namespace"
string    namespace name
```

It is possible that not all namespaces will be visible to every authenticated user. In this case the responding server will return

a subset of existing namespaces. See Security Considerations below.

Following the last "namespace" response, a status packet MUST be sent.

5. Extending Public Key Operations

In addition to adding new operations, this document describes extensions to the operations defined in RFC 4819.

5.1. Adding a Public Key

If the client wishes to add a public key, the client sends:

```
string    "add"
string    public key algorithm name
string    public key blob
boolean   overwrite
uint32    attribute-count
 string    attrib-name
 string    attrib-value
 bool      critical
repeated attribute-count times
```

This request MAY include one "namespace" attribute so that a client can save the public key into a specific namespace. It is possible for the same user to save the same key into multiple namespaces but will require multiple add requests to do so.

If the namespace appearing in an add public key request does not already exist on a server then it is created by this operation. However, if the user is not authorized to create a namespace the server MUST return SSH_PUBLICKEY_CANNOT_CREATE_NAMESPACE,

5.2. Removing a Public Key

If the client wishes to remove a public key, the client sends:

```
string    "remove"
string    public key algorithm name
string    public key blob
uint32    attribute-count
 string    attrib-name
 string    attrib-value
 bool      critical
repeated attribute-count times
```

This extension allows attributes to be added to a remove request. This request MAY include one "namespace" attribute so that a client can remove the public key from a specific namespace.

5.3. Listing Public Keys

If the client wishes to list the known public keys, the client sends:

```
string    "list"
uint32    attribute-count
 string    attrib-name
 string    attrib-value
 bool      critical
repeated attribute-count times
```

This extension allows attributes to be added to a list request. This request MAY include one "namespace" attribute so that a client can list the public keys from a specific namespace.

The server will respond with zero or more of the following responses:

```
string    "publickey"
string    public key algorithm name
string    public key blob
uint32    attribute-count
 string    attrib-name
 string    attrib-value
repeated attribute-count times
```

This response MAY include at the "namespace" attribute so that a client can tell which namespace the key resides.

6. Security Considerations

This protocol assumes that it is run over a secure channel and that the endpoints of the channel have been authenticated. Thus, this protocol assumes that it is externally protected from network-level attacks.

This protocol provides a mechanism that allows key and certificate material to be uploaded and manipulated into a server application. It is the responsibility of the server implementation to enforce access controls that may be required to limit the access allowed for any particular user to that data in a namespace. For example, one user may be allowed to list only the contents of a namespace but not add or remove keys or certificates to/from it. The server MUST return SSH_PUBLICKEY_ACTION_NOT_AUTHORIZED when a user's action goes against its defined access controls.

This protocol requires the client to assume that the server will correctly implement and observe attributes applied to keys. Implementation errors in the server could cause clients to authorize keys and certificates for access they were not intended to have, or to apply fewer restrictions than were intended.

7. IANA Considerations

Although section 3.1 defines four new status codes, these are in the 'Private Use' range of IANA's Publickey Subsystem Attributes, Registry as defubed by section 6.6.1. Conventions in [1]. No IANA action is requested for this document.

8. References

8.1. Normative References

[1] J. Galbraith, J. Van Dyke, and J. Bright, "Secure Shell Public Key Subsystem", RFC 4819, March 2007.

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

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

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

[5] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

8.1. Informative References

[6] OASIS, Key Mangement Interoperability Protocol (KMIP) 1.1, http://docs.oasis-open.org/kmip/spec/v1.1/os/kmip-spec-v1.1-os.html. 24 January 2013.

9. Authors' Addresses

Mark Joseph, PhD
P6R, Inc
1840 41st Ave
Suite 102-139
Capitola, CA 95010
US

Phone: +1 888 452 2580 (x702)
EMail: mark@p6r.com

Jim Susoy
P6R, Inc
1840 41st Ave
Suite 102-139
Capitola, CA 95010
US

Phone: +1 888 452 2580 (x701)
EMail: jim@p6r.com