Independent Submission
Request for Comments: 6042
Category: Informational
ISSN: 2070-1721

A. Keromytis
Columbia University

October 2010

Transport Layer Security (TLS) Authorization Using KeyNote

Abstract

This document specifies the use of the KeyNote trust-management system as an authorization extension in the Transport Layer Security (TLS) Handshake Protocol, according to guidelines in RFC 5878. Extensions carried in the client and server hello messages confirm that both parties support the desired authorization data types. Then, if supported by both the client and the server, KeyNote credentials are exchanged in the supplemental data handshake message.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This is a contribution to the RFC Series, independently of any other RFC stream. The RFC Editor has chosen to publish this document at its discretion and makes no statement about its value for implementation or deployment. Documents approved for publication by the RFC Editor are not a candidate for any level of Internet Standard; see 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/rfc6042.

1. Introduction

This document describes the identifiers necessary to exchange KeyNote [KEYNOTE] credential assertions inside a TLS [TLS1.0] [TLS1.1] [TLS1.2] exchange. Such credential assertions can authorize the client and/or the server to perform certain actions. In most usage scenarios, the KeyNote credential assertions will be signed by a cryptographic public key [RFC2792]. By using the X.509 key and signature encoding [X509KEY], it is possible to add KeyNote-based authorization and policy compliance support to the existing, unmodified X.509 authentication exchange in TLS.

A list of KeyNote credentials (e.g., forming a delegation chain) may be sent as part of the same payload. Alternatively, a URL [RFC3986] pointing to the location of such a list of KeyNote credentials may be provided.

In most scenarios, at least one of these credentials will be issued to the public key of the transmitter of the credentials, i.e., said public key will appear in the "Licensees" field of at least one KeyNote credential assertion. The same public key will generally be used by the transmitter of the same credentials to authenticate as part of the TLS exchange. The authentication material (e.g., cryptographic public key) that was used by the transmitter to authenticate in the TLS exchange will be provided to the KeyNote evaluation engine as an "Action Authorizer".

1.1. Conventions

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 [RFC2119].

2. KeyNote Credential Assertion Lists

The KeyNote Assertion List type definition in the TLS Authorization Data Formats registry is:

```
keynote_assertion_list(64)
```

When the keynote_assertion_list value is present, the authorization data is a list of KeyNote credential assertions that conforms to the profile in RFC 2704 [KEYNOTE].

A KeyNote assertion list is transmitted inside an AuthorizationDataEntry structure as an opaque sequence of 1 - 2^16-1 bytes:

```
opaque KeyNoteAssertionList<1..2^16-1>;
```

When KeyNoteAssertionList is used, the field contains an ASCII-encoded list of signed KeyNote assertions, as described in RFC 2704 [KEYNOTE]. The assertions are separated by two "\n" (newline) characters. A KeyNote assertion is a structure similar to a public key certificate; the main difference is that instead of a binding between a name and a public key, KeyNote assertions bind public keys to authorization rules that are evaluated by the peer when the sender later issues specific requests.

When making an authorization decision based on a list of KeyNote assertions, proper linkage between the KeyNote assertions and the public key certificate that is transferred in the TLS Certificate message is needed. Receivers of a KeyNote assertion list should initialize the ACTION_AUTHORIZER variable to be the sender's public key, which was used to authenticate the TLS exchange. If a different authentication mechanism is used, it is the responsibility of the credential issuer to issue the appropriate credentials.

3. KeyNote Credential Assertion List URL

The KeyNote Assertion List URL type definition in the TLS Authorization Data Formats registry is:

```
keynote_assertion_list_url(65)
```

A KeyNote Assertion List URL is transmitted inside an AuthorizationDataEntry structure as a URLandHash structure [AUTHZ].

When the keynote_assertion_list_url value is present, the authorization data is a list of KeyNote assertions as described in Section 2; however, the KeyNote assertion list is fetched with the supplied URL. A one-way hash value is provided to ensure that the intended KeyNote credential assertion is obtained.

Implementations that support keynote_assertion_list_url MUST support URLs that employ the HTTP scheme [HTTP]. These implementations MUST confirm that the hash value computed on the fetched authorization matches the one received in the handshake. Mismatch of the hash values SHOULD be treated as though the authorization was not provided, which will result in a bad_certificate alert [AUTHZ].

Other schemes may also be supported. When dereferencing these URLs, circular dependencies MUST be avoided. Avoiding TLS when dereferencing these URLs is one way to avoid circular dependencies. Therefore, clients using the HTTP scheme MUST NOT use these TLS extensions if the Upgrade mechanism in HTTP [UPGRADE] is used. For other schemes, similar care must be taken to avoid using these TLS extensions.

4. IANA Considerations

With this document, IANA has registered two new entries in the TLS Authorization Data Formats registry: keynote_assertion_list(64) and keynote_assertion_list_url(65). This registry is defined in [AUTHZ].

5. Security Considerations

There are no security considerations beyond those discussed in [KEYNOTE], [RFC2792], and [AUTHZ].

6. References

6.1. Normative References

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

[TLS1.0] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[TLS1.1] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[TLS1.2] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[HTTP] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[UPGRADE] Khare, R. and S. Lawrence, "Upgrading to TLS Within HTTP/1.1", RFC 2817, May 2000.

[KEYNOTE] Blaze, M., Feigenbaum, J., Ioannidis, J., and A. Keromytis, "The KeyNote Trust-Management System Version 2", RFC 2704, September 1999.

[AUTHZ] Brown, M. and R. Housley, "Transport Layer Security (TLS) Authorization Extensions", RFC 5878, May 2010.

6.2. Informative References

[RFC2792] Blaze, M., Ioannidis, J., and A. Keromytis, "DSA and RSA Key and Signature Encoding for the KeyNote Trust Management System", RFC 2792, March 2000.

[X509KEY] Keromytis, A., "X.509 Key and Signature Encoding for the KeyNote Trust Management System", RFC 5708, January 2010.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

Appendix A. Updated TLS Authorization Data Structures

For clarity, this Appendix shows an updated version of the relevant data structures from Section 3.3 in [AUTHZ] with the new entries described in this document. The added elements are denoted with two asterisks ("**") at the end of the respective lines.

```
struct {
   AuthorizationDataEntry authz_data_list<1..2^16-1>;
} AuthorizationData;

struct {
   AuthzDataFormat authz_format;
   select (AuthzDataFormat) {
      case x509_attr_cert:              X509AttrCert;
      case saml_assertion:              SAMLAssertion;
      case x509_attr_cert_url:          URLandHash;
      case saml_assertion_url:          URLandHash;
      case keynote_assertion_list:      KeyNoteAssertionList;   **
      case keynote_assertion_list_url:  URLandHash;             **
   }
} AuthorizationDataEntry;

enum {
   x509_attr_cert(0), saml_assertion(1), x509_attr_cert_url(2),
   saml_assertion_url(3),
   keynote_assertion_list(64), keynote_assertion_list_url(65),  **
   (255)
} AuthzDataFormat;

opaque X509AttrCert<1..2^16-1>;

opaque SAMLAssertion<1..2^16-1>;

opaque KeyNoteAssertionList<1..2^16-1>;                         **

struct {
   opaque url<1..2^16-1>;
   HashAlgorithm hash_alg;
   select (hash_alg) {
      case md5:    MD5Hash;
      case sha1:   SHA1Hash;
      case sha224: SHA224Hash;
      case sha256: SHA256Hash;
      case sha384: SHA384Hash;
      case sha512: SHA512Hash;
   } hash;
} URLandHash;
```

```
      enum {
         none(0), md5(1), sha1(2), sha224(3), sha256(4), sha384(5),
         sha512(6), (255)
      } HashAlgorithm;

      opaque MD5Hash[16];

      opaque SHA1Hash[20];

      opaque SHA224Hash[28];

      opaque SHA256Hash[32];

      opaque SHA384Hash[48];

      opaque SHA512Hash[64];
```

Author's Address

Angelos D. Keromytis
Department of Computer Science
Columbia University
Mail Code 0401
1214 Amsterdam Avenue
New York, NY 10027
USA

EMail: angelos@cs.columbia.edu