TLS Working Group
Internet-Draft
Expires: July 21, 2002

N. Mavroyanopoulos
January 22, 2002

Using OpenPGP keys for TLS authentication

<draft-ietf-tls-openpgp-keys-00.txt>

Status of this Memo

Abstract

This document builds upon the TLS Protocol Specification [TLS]. The extensions described herein are intended to apply to Version 1.0 of the TLS specification.

The purpose of this document is to update the TLS protocol with extensions to support the certificates, public key algorithms, symmetric ciphers, hash algorithms, and trust model used by OpenPGP [OpenPGP].

This document uses the same notation used in the TLS Protocol draft.

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.

1. Introduction

At the time of writing, TLS [TLS] uses X.509 digital certificiates, or Kerberos, for authentication.

OpenPGP keys (sometimes called OpenPGP certificates), provide security services for electronic communications. This document will will update the TLS protocol with extensions to support these keys.

2. OpenPGP keys for TLS authentication

The X.509 [X509] certificates recommended for use with TLS will not be used in conjunction with OpenPGP keys. An implementation SHOULD be able to support both TLS with X.509 certificates and TLS with OpenPGP keys. Implementations are not required to support both. The "peer certificate" in the session state of TLS MAY refer to either X.509 or OpenPGP.

2.1 Changes to the Handshake Message Contents

This section describes the changes to the TLS handshake message contents when OpenPGP keys are to be used for authentication.

2.1.1 Hello Messages

2.1.1.1 Extension Type

A new value, "cert_type(7)", has been added to the enumerated ExtensionType, defined in [TLSEXT]. This value is used as the extension number for the extensions in both the client hello message and the server hello message. This value was chosen based on the version of defined in [TLSEXT] that was current at the time of writing, so may be changed in future.

2.1.1.2 Client Hello

An extension of type CertificateTypeExtension is appended to the standard client hello message using the client hello extension mechanism defined in [TLSEXT].

This extension carries a list of supported Certificate types the client can use, sorted by client preference. This extension SHOULD NOT be used if the client supports only X.509 certificates.

```
enum { client, server } ClientOrServerExtension;

enum { X.509(0), OpenPGP(1), (255) } CertificateType;
```

```
struct {
   select(ClientOrServerExtension) {
      case client:
         CertificateType certificate_type<1..2^8-1>;
      case server:
         CertificateType certificate_type;
   }
} CertificateTypeExtension;
```

2.1.1.3 Server Hello

The certificate type selected by the server (certificate_type), is appended to the server hello message using the hello extension mechanism defined in [TLSEXT].

The certificate type selected by the server (certificate_type), is encoded in an CertificateTypeExtension structure, which is sent in an extended server hello message, using an extension of type "cert_type".

The CertificateTypeExtension structure may be omited if the server only supports X.509 certificates. In case the server does not support any of the certificate types sent by the client, the server should terminate the connection with a fatal alert of "unsupported_certificate" type.

2.1.2 Server certificate

The contents of the certificate message sent from server to client and vice versa are determined by the negotiated certificate type and the selected cipher suite's key exchange algorithm.

In case OpenPGP certificate type is negotiated then it is required to present an OpenPGP key in the Certificate message. The OpenPGP key must contain a public key that matches the selected key exchange algorithm, as shown below.

Key Exchange Algorithm	OpenPGP Key Type
RSA	RSA public key which can be used for encryption.
DHE_DSS	DSS public key.
DHE_RSA	RSA public key which can be used for signing.

An OpenPGP key appearing in the Certificate message will be sent in binary OpenPGP format. The option is also available to send an OpenPGP fingerprint, instead of sending the entire key. The process of fingerprint generation is described in [OpenPGP]. The peer will respond with a "certificate_unobtainable" fatal alert if the key with the given key fingerprint cannot be found. The "certificate_unobtainable" fatal alert is defined in section 4 of [TLSEXT].

If the key is not valid, expired, revoked, corrupt, the appropriate fatal alert message is sent from section A.3 of the TLS specification. If a key is valid and neither expired nor revoked, it is accepted by the protocol. The validation procedure is a local matter ouside the scope of this document.

```
enum {
     key_fingerprint (0), key (1), (255)
} PGPKeyDescriptorType;

opaque PGPKeyFingerprint<16..20>;

opaque PGPKey<0..2^24-1>

struct {
     PGPKeyDescriptorType descriptorType;
     select (descriptorType) {
          case key_fingerprint: PGPKeyFingerprint;
          case key: PGPKey;
     }
```

```
   } Certificate;
```

2.1.3 Certificate request

The semantics of this message remain the same as in the TLS specification. However the structure of this message has been modified for OpenPGP keys.

```
   enum {
       rsa_sign(1), dss_sign(2), (255)
   } ClientCertificateType;

   struct {
       ClientCertificateType certificate_types<1..2^8-1>;
   } CertificateRequest;
```

certificate_types is a list of the types of certificates requested, sorted in order of the server's preference.

2.1.4 Client certificate

The client certificate message is sent using the same formatting as the server certificate message. This message is only sent in response to the certificate request message. If no OpenPGP key is available from the client, then a certificate that contains an empty PGPKey is returned. The server may respond with a "handshake_failure" fatal

alert if client authentication is required. This transaction follows the TLS specification.

2.1.5 Server key exchange

The server key exchange message for OpenPGP keys is identical to the TLS specification.

2.1.6 Certificate Verify

The certificate verify message for OpenPGP key types is identical to the TLS specification.

2.1.7 Finished

The finished message for OpenPGP key types is identical to the description in the specification.

3. Cipher suites

No new cipher suites are required to use OpenPGP keys. OpenPGP keys can be combined with existing cipher suites defined in [TLS], except the ones marked as "Exportable". Exportable cipher suites SHOULD NOT be used with OpenPGP keys.

3.1 New cipher suites

Some additional cipher suites are defined here in order to support algorithms which are defined in [OpenPGP] but are not present in [TLS].

```
CipherSuite TLS_DHE_DSS_WITH_CAST_128_CBC_SHA      = { 0x00, 0x70 };
CipherSuite TLS_DHE_DSS_WITH_CAST_128_CBC_RMD      = { 0x00, 0x71 };
CipherSuite TLS_DHE_DSS_WITH_3DES_EDE_CBC_RMD      = { 0x00, 0x72 };
CipherSuite TLS_DHE_DSS_WITH_AES_128_CBC_RMD       = { 0x00, 0x73 };
CipherSuite TLS_DHE_DSS_WITH_AES_256_CBC_RMD       = { 0x00, 0x74 };

CipherSuite TLS_DHE_RSA_WITH_CAST_128_CBC_SHA      = { 0x00, 0x75 };
CipherSuite TLS_DHE_RSA_WITH_CAST_128_CBC_RMD      = { 0x00, 0x76 };
CipherSuite TLS_DHE_RSA_WITH_3DES_EDE_CBC_RMD      = { 0x00, 0x77 };
CipherSuite TLS_DHE_RSA_WITH_AES_128_CBC_RMD       = { 0x00, 0x78 };
CipherSuite TLS_DHE_RSA_WITH_AES_256_CBC_RMD       = { 0x00, 0x79 };

CipherSuite TLS_RSA_WITH_CAST_128_CBC_SHA          = { 0x00, 0x7A };
CipherSuite TLS_RSA_WITH_CAST_128_CBC_RMD          = { 0x00, 0x7B };
CipherSuite TLS_RSA_WITH_3DES_EDE_CBC_RMD          = { 0x00, 0x7C };
CipherSuite TLS_RSA_WITH_AES_128_CBC_RMD           = { 0x00, 0x7D };
CipherSuite TLS_RSA_WITH_AES_256_CBC_RMD           = { 0x00, 0x7E };
```

All of the above cipher suites use either the CAST [CAST], AES [AES], or 3DES block ciphers in CBC mode. The choice of hash

is either SHA-1 or RIPEMD-160. Implementations are not required to support the above cipher suites.

References

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

[OpenPGP] Callas, J., Donnerhacke, L., Finney, H., Thayer, R., "OpenPGP Message Format", RFC 2440, November 1998.

[TLSEXT] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J. and Wright, T., "TLS Extensions", work in progress, December 2001.

[X509] CCITT. Recommendation X.509: "The Directory - Authentication Framework". 1988.

[CAST] Adams, C., "The CAST-128 Encryption Algorithm", RFC 2144, May 1997.

[AES] J. Daemen, V. Rijmen, "The Rijndael Block Cipher" http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf 3rd September 1999.

Author's Address

Nikos Mavroyanopoulos
nmav@gnu.org

Full Copyright Statement

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

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