INTERNET-DRAFT   Matthew Hur
Transport Layer Security Working Group   Cisco Systems
draft-ietf-tls-kerb-00.txt   Ari Medvinsky
Obsoletes: RFC 2712   Keen.com
November 06, 2000 (Expires May 06, 2001)

Kerberos Cipher Suites in Transport Layer Security (TLS)

0. Status Of this Memo

1. Abstract

RFC 2712 [KERBTLS] introduced mechanisms for supporting Kerberos [KERB] authentication within the TLS protocol [TLS]. This document extends RFC 2712 to support delegation of Kerberos credentials. In this way, a TLS server may obtain a Kerberos service ticket on behalf of the TLS client. Thus, a single client identity may be used for authentication within a multi-tier architecture. This draft also proposes a mechanism for a TLS server to indicate Kerberos-specific

information to the client within the certificate request message in the initial exchange.

2. Introduction

Flexibility is one of the main strengths of the TLS protocol. Clients and servers can negotiate cipher suites to meet specific security and administrative policies. RFC 2712 specified how TLS could be extended to support organizations with heterogeneous security deployments that include authentication systems based on symmetric cryptography. Kerberos, originally developed at MIT, is based on an open standard and is the most widely deployed symmetric key authentication system. Just as other documents specify hybrid asymmetric/symmetric key protocols [PKINIT] [PKCROSS] [PKTAPP], this document specifies how TLS may incorporate both symmetric and asymmetric key crypto systems.

This document describes the use of Kerberos authentication within the TLS framework. This achieves mutual authentication and the establishment of a master secret using Kerberos credentials. Additionally, this document specifies support for delegation of Kerberos credentials, which enables end to end authentication within an n-tier architecture. The proposed changes are minimal and, in fact, no different from adding a new public key algorithm to the TLS framework.

3. Kerberos Authentication Option In TLS

This section describes the addition of the Kerberos authentication option to the TLS protocol. Throughout this document, we refer to the basic SSL handshake shown in Figure 1. For a review of the TLS handshake see [TLS].

```
+---------------------------------------------------------------------+
| CLIENT                                             SERVER           |
| ------                                             ------           |
| ClientHello                                                         |
|                   ---------------------------->                     |
|                                                    ServerHello      |
|                                                    Certificate *    |
|                                                    ServerKeyExchange* |
|                                                    CertificateRequest* |
|                                                    ServerHelloDone  |
|                   <----------------------------                     |
| Certificate*                                                        |
| ClientKeyExchange                                                   |
| CertificateVerify*                                                  |
| change cipher spec                                                  |
| Finished                                                            |
```

```
|     |              --------------------------->                   |
|     |                                            change cipher spec |
|     |                                            Finished           |
|     |                                                |              |
|     |                                                |              |
| Application Data <--------------------------->  Application Data    |
+---------------------------------------------------------------------+
```

FIGURE 1: The TLS protocol. All messages followed by a star are optional. Note: This figure was taken from RFC 2246.

The TLS security context is negotiated in the client and server hello messages. For example: TLS_RSA_WITH_RC4_MD5 means the initial authentication will be done using the RSA public key algorithm, RC4 will be used for the session key, and MACs will be based on the MD5 algorithm. Thus, to facilitate the Kerberos authentication option, we must start by defining Kerberos cipher suites including (but not limited to):

```
CipherSuite     TLS_KRB5_WITH_DES_CBC_SHA            = { 0x00,0x1E };
CipherSuite     TLS_KRB5_WITH_3DES_EDE_CBC_SHA       = { 0x00,0x1F };
CipherSuite     TLS_KRB5_WITH_RC4_128_SHA            = { 0x00,0x20 };
CipherSuite     TLS_KRB5_WITH_IDEA_CBC_SHA           = { 0x00,0x21 };
CipherSuite     TLS_KRB5_WITH_DES_CBC_MD5            = { 0x00,0x22 };
CipherSuite     TLS_KRB5_WITH_3DES_EDE_CBC_MD5       = { 0x00,0x23 };
CipherSuite     TLS_KRB5_WITH_RC4_128_MD5            = { 0x00,0x24 };
CipherSuite     TLS_KRB5_WITH_IDEA_CBC_MD5           = { 0x00,0x25 };

CipherSuite     TLS_KRB5_EXPORT_WITH_DES_CBC_40_SHA  = { 0x00,0x26 };
CipherSuite     TLS_KRB5_EXPORT_WITH_RC2_CBC_40_SHA  = { 0x00,0x27 };
CipherSuite     TLS_KRB5_EXPORT_WITH_RC4_40_SHA      = { 0x00,0x28 };
CipherSuite     TLS_KRB5_EXPORT_WITH_DES_CBC_40_MD5  = { 0x00,0x29 };
CipherSuite     TLS_KRB5_EXPORT_WITH_RC2_CBC_40_MD5  = { 0x00,0x2A };
CipherSuite     TLS_KRB5_EXPORT_WITH_RC4_40_MD5      = { 0x00,0x2B };

CipherSuite     TLS_KRB5_WITH_NULL_SHA               = { 0x00,0x?? };
CipherSuite     TLS_KRB5_WITH_NULL_MD5               = { 0x00,0x?? };
CipherSuite     TLS_KRB5_WITH_NULL_NULL              = { 0x00,0x?? };
```

To establish a Kerberos-based security context, one or more of the above cipher suites must be specified in the client hello message. If the TLS server supports the Kerberos authentication option, the server hello message, sent to the client, will confirm the Kerberos cipher suite selected by the server. The server's certificate and the ServerKeyExchange shown in Figure 1 will be omitted since authentication and the establishment of a master secret will be done using the client's Kerberos credentials for the TLS server. Note that these messages are specified as optional in the TLS protocol; therefore, omitting them is permissible.

The Kerberos option affects three of the TLS messages: the CertificateRequest, the client Certificate, and the

ClientKeyExchange. However, only the client Certificate and the ClientKeyExchange are required.

3.1. Usage of the CertificateRequest Message

If the server accepts a Kerberos-based ciphersuite, then it MUST send the CertificateRequest message to the client. This message conveys Kerberos-specific characteristics such as realm name or attributes such as forwarded ticket.

RFC 2246 defines the CertificateRequest message as follows:

```
+-------------------------------------------------------------------+
|                                                                   |
|   enum {                                                          |
|       rsa_sign(1), dss_sign(2), rsa_fixed_dh(3), dss_fixed_dh(4), |
|       (255)                                                       |
|   } ClientCertificateType;                                        |
|                                                                   |
|   opaque DistinguishedName<1..2^16-1>;                            |
|                                                                   |
|   struct { ClientCertificateType certificate_types<1..2^8-1>;     |
|            DistinguishedName certificate_authorities<3..2^16-1>;  |
|   } CertificateRequest;                                           |
|                                                                   |
+-------------------------------------------------------------------+
```

FIGURE 2: CertificateRequest message from RFC 2246

This specification defines a new ClientCertificateType for a Kerberos certificate. This enables a client to respond to the CertificateRequest message when using Kerberos ciphersuites. Thus the following change for ClientCertificateType is required (Figure 3).

```
+-------------------------------------------------------------------+
|                                                                   |
|   enum {                                                          |
|       rsa_sign(1), dss_sign(2), rsa_fixed_dh(3), dss_fixed_dh(4), |
|       kerberos(5), (255)                                          |
|   } ClientCertificateType;                                        |
|                                                                   |
+-------------------------------------------------------------------+
```

FIGURE 3: New Kerberos ClientCertificateType

In the case of a public key based authentication algorithm, the opaque DistinguishedName field is derived from [X509], and it contains the name of an acceptable certification authority (This is as specified in [TLS]). In the case of a Kerberos ClientCertificateType, the DistinguishedName field is defined to represent Kerberos information (KerbInfo) as shown in Figure 4.

```
+------------------------------------------------------------------+
|                                                                  |
|   enum                                                           |
|   {                                                              |
|      srv_tkt(1), fwd_tgt(2), (255)                               |
|   } KerbInfoType;                                                |
|                                                                  |
|   enum                                                           |
|   {                                                              |
|      initial_tkt_required(1), (255)                              |
|   } AttrType; /* This may be extended to include attributes    */ |
|               /*  such as forwardable or renewable for example */ |
|                                                                  |
|   struct                                                         |
|   {                                                              |
|      AttrType       attr_type;                                   |
|      opaque         attr_data <0..2^16-1>;                       |
|   } AttrInfoType                                                 |
|                                                                  |
|   struct                                                         |
|   {                                                              |
|      uint32         length; /* length of this struct */          |
|      KerbInfoType   type;                                        |
|      opaque         sname <0..2^16-1>;                           |
|      opaque         srealm <0..2^16-1>;                          |
|      opaque         cname <0..2^16-1>;                           |
|      opaque         crealm <0..2^16-1>;                          |
|      AttrInfoType   attr_info <0..2^16-1>; /* sequence of      */ |
|                                            /* attributes       */ |
|      uint32         etypes <0..2^16-1>; /* list of supported   */ |
|                                         /* Kerberos etypes     */ |
|                                         /* for authentication  */ |
|   } TktInfo;                                                     |
|                                                                  |
|   struct                                                         |
|   {                                                              |
|      uint16        num; /* number of TktInfo structs */          |
|      TktInfo       tkt_info <1..2^20-1>; /* MUST have at least */ |
|                                          /* 1 TktInfo structs */  |
|   } KerbInfo                                                     |
|                                                                  |
+------------------------------------------------------------------+
```

FIGURE 4: Kerberos Information for CertificateRequest Message

3.2. Usage of the Client Certificate Message

As specified by [TLS], when the client receives the CertificateRequest message, it MUST respond with the client Certificate message. As stated above, this specification defines a Kerberos certificate type. The format for the Kerberos certificate is specified in figure 5 below. This structure consists of a Kerberos AP-REQ message that is used for authenticating the client to he server. It optionally contains a series of Kerberos KRB-CRED messages to convey delegated credentials.

Note that the client may determine the type of credentials to send to the server, based on local policy. Part of the input to a client's decision may come from the Kerberos KDC. For example, The client may convey a delegated ticket based on the ok-as-delegate ticket flag set in the service ticket.

```
+-------------------------------------------------------------------+
|                                                                   |
|  opaque KrbCred <1..2^16-1>; /* Kerberos-defined KRB-CRED */      |
|                                                                   |
|  struct                                                           |
|  {                                                                |
|      opaque    ap_req <1..2^16-1>;                                |
|      uint16    num; /* number of KrbCred structs */               |
|      KrbCred   krb_cred <0..2^20-1>;                              |
|  } KerberosCert;                                                  |
|                                                                   |
+-------------------------------------------------------------------+
```

FIGURE 5: Kerberos Certificate Type

3.3. Usage of the ClientKeyExchange Message

The Kerberos option must be added to the ClientKeyExchange message as shown in Figure 6.

```
+-------------------------------------------------------------------+
|                                                                   |
|  struct                                                           |
|  {                                                                |
|      select (KeyExchangeAlgorithm)                                |
|      {                                                            |
|          case krb:             KerbEncryptedPreMasterSecret;      |
|          case rsa:             EncryptedPreMasterSecret;          |
|          case diffie_hellman:  ClientDiffieHellmanPublic;         |
|      } Exchange_keys;                                             |
|  } ClientKeyExchange;                                             |
|                                                                   |
| KerbEncryptedPreMasterSecret contains the PreMasterSecret         |
| encrypted within a Kerberos-defined EncryptedData structure.      |
| The encryption key is sealed in the ticket sent in the Client     |
| Certificate message.                                              |
|                                                                   |
+-------------------------------------------------------------------+
```

FIGURE 6: The Kerberos option in the ClientKeyExchange.

To use the Kerberos authentication option, the TLS client must obtain a service ticket for the TLS server. In TLS, the ClientKeyExchange message is used to pass a random 48-byte pre-master secret to the server.

The client and server then use the pre-master secret to independently derive the master secret, which in turn is used for generating session keys and for MAC computations. Thus, if the Kerberos option is selected, the pre-master secret structure is the same as that used in the RSA case; it is encrypted under the Kerberos session key and sent to the TLS server along with the Kerberos credentials (see Figure 2). The ticket and authenticator are encoded per RFC 1510 (ASN.1 encoding). Once the ClientKeyExchange message is received, the server's secret key is used to unwrap the credentials and extract the pre-master secret.

Lastly, the client and server exchange the finished messages to complete the handshake. At this point we have achieved the following:

1) A master secret, used to protect all subsequent communication, is securely established.

2) Mutual client-server authentication is achieved, since the TLS server proves knowledge of the master secret in the finished message.

Kerberos fits seamlessly into TLS, without adding any new messages.

4. Naming Conventions:

To obtain an appropriate service ticket, the TLS client must determine the principal name of the TLS server. The Kerberos service naming convention is used for this purpose, as follows:

```
host/MachineName@Realm
 where:
   - The literal, "host", follows the Kerberos convention when not
     concerned about the protection domain on a particular machine.
   - "MachineName" is the particular instance of the service.
   - The Kerberos "Realm" is the domain name of the machine.
```

As specified above, in the CertificateRequest message, the server may indicate the appropriate principal name and realm.

5. Summary

The proposed Kerberos authentication option is added in exactly the same manner as a new public key algorithm would be added to TLS. Furthermore, it establishes the master secret in exactly the same manner.

6. Security Considerations

Kerberos ciphersuites are subject to the same security considerations as the TLS protocol. In addition, just as a public key implementation must take care to protect the private key (for example the PIN for a smartcard), a Kerberos implementation must take care to protect the long lived secret that is shared between the principal and the KDC. In particular, a weak password may be subject to a dictionary attack. In order to strengthen the initial authentication to a KDC, an implementor may choose to utilize secondary authentication via a token card, or one may utilize initial authentication to the KDC based on public key cryptography (commonly known as PKINIT - a product of the Kerberos working group of the IETF).

The unauthenticated CertificateRequest message, specified above, enables the server to request a particular client principal name as well as a particular service principal name. In the event that a service principal name is specified, there is a risk that the client may be tricked into requesting a ticket for a rogue server. Furthermore, if delegation is requested, the client may be tricked into forwarding its TGT to a rogue server. In order to assure that a service ticket is obtained for the correct server, the client should rely on a combination of its own local policy, local configuration information, and information supplied by the KDC. The client may choose to use only the naming convention specified in section 4. The client may rely on the KDC performing name cannonicalization (this is a matter that is adressed in revisions to RFC 1510).

The client must apply its local policy to determine whether or not to forward its credentials. As previously stated, the client should incorporate information from the KDC, in particular the ok-as-delegate ticket flag, in making such a policy decision.

A forwarded TGT presents more vulnerabilities in the event of a rogue server or the compromise of the session key. An attacker would be able to impersonate the client to obtain new service tickets. Such an attack may be mitigated by the use of restrictions, such as those described in [Neuman].

7. Acknowledgements

We would like to thank the following people for their input for this document:
Clifford Neuman from ISI

John Brezak and David Mowers from Microsoft

8. References

[KERBTLS] A. Medvinsky and M. Hur, "Addition of Kerberos Cipher Suites to Transport Layer Security (TLS)", RFC 2712, October 1999.

[KERB] J. Kohl and C. Neuman, "The Kerberos Network Authentication Service (V5)", RFC 1510, September 1993.

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

[PKINIT] B. Tung, C. Neuman, M. Hur, A. Medvinsky, S. Medvinsky, J. Wray, J. Trostle. Public Key Cryptography for Initial Authentication in Kerberos. draft-ietf-cat-kerberos-pk-init-12.txt

[PKTAPP] A. Medvinsky, M. Hur, S. Medvinsky, C. Neuman. Public Key Utilizing Tickets for Application Servers (PKTAPP). draft-ietf-cat-kerberos-pk-tapp-03.txt

[PKCROSS] M. Hur, B. Tung, T. Ryutov, C. Neuman, G. Tsudik, A. Medvinsky, B. Sommerfeld. Public Key Cryptography for Cross-Realm Authentication in Kerberos. draft-ietf-cat-kerberos-pk-cross-06.txt

[X509] ITU-T (formerly CCITT) Information technology - Open Systems Interconnection - The Directory: Authentication Framework Recommendation X.509 ISO/IEC 9594-8

[NEUMAN] B.C. Neuman, "Proxy-Based Authorization and Accounting for Distributed Systems". Proceedings of the 13th International Conference on Distributed Computing Systems, May 1993

9. Authors' Addresses

Matthew Hur
Cisco Systems
500 108th Ave. NE, Suite 500
Bellevue, WA 98004
Phone:
EMail: mhur@cisco.com
http://www.cisco.com

Ari Medvinsky
Keen.com

2480 Sand Hill Road, Suite 200
Menlo Park, CA 94025
Phone: +1 415 284 4085
EMail: ari@keen.com
http://www.keen.com

10. Full Copyright Statement

Appendices

A. Changes from RFC 2712

Added new cipher suites with NULL confidentiality:

```
 TLS_KRB5_WITH_NULL_SHA
 TLS_KRB5_WITH_NULL_MD5
 TLS_KRB5_WITH_NULL_NULL
```

RFC 2712 utilized only the ClientKeyExchange message for conveying the Kerberos credentials and encrypted premaster-secret. This specification moves the Kerberos credentials to the client certificate message, and it allows the client to pass delegated credentials as well. Additionally, this specification allows the server to specify Kerberos-specific information (realm, delegation required, etc.) in the CertificateRequest message.

B. IESG Note from RFC 2712

The 40-bit ciphersuites defined in this memo are included only for the purpose of documenting the fact that those ciphersuite codes have already been assigned. 40-bit ciphersuites were designed to comply with US-centric, and now obsolete, export restrictions. They were never secure, and nowadays are inadequate even for casual applications. Implementation and use of the 40-bit ciphersuites defined in this document, and elsewhere, is strongly discouraged.