        Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)
                              Freshness Extension

Abstract

   This document describes how to further extend the Public Key
   Cryptography for Initial Authentication in Kerberos (PKINIT)
   extension (defined in RFC 4556) to exchange an opaque data blob that
   a Key Distribution Center (KDC) can validate to ensure that the
   client is currently in possession of the private key during a PKINIT
   Authentication Service (AS) exchange.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 7841.

   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/rfc8070.

Table of Contents

1. Introduction

   The Kerberos PKINIT extension [RFC4556] defines two schemes for using
   asymmetric cryptography in a Kerberos pre-authenticator. One uses
   Diffie-Hellman key exchange and the other depends on public key
   encryption. The public key encryption scheme is less commonly used
   for two reasons:

   o Elliptic Curve Cryptography (ECC) Support for PKINIT [RFC5349]
      only specified Elliptic Curve Diffie-Hellman (ECDH) key agreement,
      so it cannot be used for public key encryption.

   o Public key encryption requires certificates with an encryption
      key, which is not deployed on many existing smart cards.

   In the Diffie-Hellman exchange, the client uses its private key only
   to sign the AuthPack structure (specified in Section 3.2.1 of
   [RFC4556]), which is performed before any traffic is sent to the KDC.
   Thus, a client can generate requests with future times in the
   PKAuthenticator, and then send those requests at those future times.
   Unless the time is outside the validity period of the client's
   certificate, the KDC will validate the PKAuthenticator and return a
   Ticket-Granting Ticket (TGT) the client can use without possessing
   the private key.

As a result, a client performing PKINIT with the Diffie-Hellman key
exchange does not prove current possession of the private key being
used for authentication. It proves only prior use of that key.
Ensuring that the client has current possession of the private key
requires that the signed PKAuthenticator data include information
that the client could not have predicted.

1.1. Kerberos Message Flow Using KRB_AS_REQ without Pre-authentication

Today, password-based AS exchanges [RFC4120] often begin with the
client sending a KRB_AS_REQ without pre-authentication. When the
principal requires pre-authentication, the KDC responds with a
KRB_ERROR containing information needed to complete an AS exchange,
such as the supported encryption types and salt values. This message
flow is illustrated below:

```
Client                                     KDC

AS-REQ without pre-authentication    ---->
                                     <----     KRB-ERROR

AS-REQ                               ---->
                                     <----     AS-REP

TGS-REQ                              ---->
                                     <----     TGS-REP
```

                        Figure 1

We can use a similar message flow with PKINIT, allowing the KDC to
provide a token for the client to include in its KRB_AS_REQ to ensure
that the PA_PK_AS_REQ [RFC4556] was not pre-generated.

1.2. Requirements Language

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. Message Exchanges

   The following summarizes the message flow with extensions to
   [RFC4120] and [RFC4556] required to support a KDC-provided freshness
   token during the initial request for a ticket:

   1. The client generates a KRB_AS_REQ, as specified in Section 2.9.3
       of [RFC4120], that contains no PA_PK_AS_REQ and includes a
       freshness token request.

   2. The KDC generates a KRB_ERROR, as specified in Section 3.1.4 of
       [RFC4120], providing a freshness token.

   3. The client receives the error, as specified in Section 3.1.5 of
       [RFC4120], extracts the freshness token, and includes it as part
       of the KRB_AS_REQ as specified in [RFC4120] and [RFC4556].

   4. The KDC receives and validates the KRB_AS_REQ, as specified in
       Section 3.2.2 of [RFC4556], then additionally validates the
       freshness token.

   5. The KDC and client continue, as specified in [RFC4120] and
       [RFC4556].

2.1. Generation of KRB_AS_REQ Message

   The client indicates support of freshness tokens by adding a padata
   element with padata-type PA_AS_FRESHNESS and padata-value of an empty
   octet string.

2.2. Generation of KRB_ERROR Message

   The KDC will respond with a KRB_ERROR [RFC4120] message with the
   error-code KDC_ERR_PREAUTH_REQUIRED [RFC4120] adding a padata element
   with padata-type PA_AS_FRESHNESS and padata-value of the freshness
   token to the METHOD-DATA object.

2.3. Generation of KRB_AS_REQ Message

   After the client receives the KRB-ERROR message containing a
   freshness token, it extracts the PA_AS_FRESHNESS padata-value field
   of the PA-DATA structure as an opaque data blob. The PA_AS_FRESHNESS
   padata-value field of the PA-DATA structure SHALL then be added as an
   opaque blob in the freshnessToken field when the client generates the
   PKAuthenticator specified in Section 4 for the PA_PK_AS_REQ message.
   This ensures that the freshness token value will be included in the
   signed data portion of the KRB_AS_REQ value.

2.4. Receipt of KRB_AS_REQ Message

   If the realm requires freshness and the PA_PK_AS_REQ message does not
   contain the freshness token, the KDC MUST return a KRB_ERROR
   [RFC4120] message with the error-code KDC_ERR_PREAUTH_FAILED
   [RFC4120] with a padata element with padata-type PA_AS_FRESHNESS and
   padata-value of the freshness token to the METHOD-DATA object.

   When the PA_PK_AS_REQ message contains a freshness token, after
   validating the PA_PK_AS_REQ message normally, the KDC will validate
   the freshnessToken value in the PKAuthenticator in an implementation-
   specific way. If the freshness token is not valid, the KDC MUST
   return a KRB_ERROR [RFC4120] message with the error-code
   KDC_ERR_PREAUTH_EXPIRED [RFC6113]. The e-data field of the error
   contains a METHOD-DATA object [RFC4120], which specifies a valid
   PA_AS_FRESHNESS padata-value. Since the freshness tokens are
   validated by KDCs in the same realm, standardizing the contents of
   the freshness token is not a concern for interoperability.

2.5. Receipt of Second KRB_ERROR Message

   If a client receives a KDC_ERR_PREAUTH_EXPIRED KRB_ERROR message that
   includes a freshness token, it SHOULD retry using the new freshness
   token.

3. PreAuthentication Data Types

   The following are the new PreAuthentication data types:

              +----------------------+------------------+
              | Padata and Data Type | Padata-type Value |
              +----------------------+------------------+
              | PA_AS_FRESHNESS | 150 |
              +----------------------+------------------+

4. Extended PKAuthenticator

   The PKAuthenticator structure specified in Section 3.2.1 of [RFC4556]
   is extended to include a new freshnessToken as follows:

   PKAuthenticator ::= SEQUENCE {
      cusec   [0] INTEGER (0..999999),
      ctime   [1] KerberosTime,
                    -- cusec and ctime are used as in [RFC4120], for
                    -- replay prevention.
      nonce   [2] INTEGER (0..4294967295),
                    -- Chosen randomly; this nonce does not need to
                    -- match with the nonce in the KDC-REQ-BODY.
      paChecksum   [3] OCTET STRING OPTIONAL,
                    -- MUST be present.
                    -- Contains the SHA1 checksum, performed over
                    -- KDC-REQ-BODY.
      ...,
      freshnessToken   [4] OCTET STRING OPTIONAL,
                    -- PA_AS_FRESHNESS padata value as received from the
                    -- KDC. MUST be present if sent by KDC
      ...
   }

5. IANA Considerations

   IANA has assigned numbers for PA_AS_FRESHNESS listed in a subregistry
   of the "Kerberos Parameters" registry titled "Pre-authentication and
   Typed Data" as follows:

                +------+-----------------+-----------+
                | Type | Value | Reference |
                +------+-----------------+-----------+
                | 150 | PA_AS_FRESHNESS | [RFC8070] |
                +------+-----------------+-----------+

6. Security Considerations

   The freshness token SHOULD include signing, encrypting, or sealing
   data from the KDC to determine authenticity and prevent tampering.

   Freshness tokens serve to guarantee that the client had the key when
   constructing the AS-REQ. They are not required to be single use
   tokens or bound to specific AS exchanges. Part of the reason the
   token is opaque is to allow KDC implementers the freedom to add
   additional functionality as long as the tokens expire so that the
   "freshness" guarantee remains.

7. Interoperability Considerations

   Since the client treats the KDC-provided data blob as opaque,
   changing the contents will not impact existing clients. Thus,
   extensions to the freshness token do not impact client
   interoperability.

   Clients SHOULD NOT reuse freshness tokens across multiple exchanges.
   There is no guarantee that a KDC will allow a once-valid token to be
   used again. Thus, clients that do not retry with a new freshness
   token may not be compatible with KDCs, depending on how they choose
   to implement freshness validation.

   Since upgrading clients takes time, implementers may consider
   allowing both freshness-token based exchanges and "legacy" exchanges
   without use of freshness tokens. However, until freshness tokens are
   required by the realm, the existing risks of pre-generated
   PKAuthenticators will remain.

8. Normative References

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119,
              DOI 10.17487/RFC2119, March 1997,
              <http://www.rfc-editor.org/info/rfc2119>.

   [RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The
              Kerberos Network Authentication Service (V5)", RFC 4120,
              DOI 10.17487/RFC4120, July 2005,
              <http://www.rfc-editor.org/info/rfc4120>.

   [RFC4556] Zhu, L. and B. Tung, "Public Key Cryptography for Initial
              Authentication in Kerberos (PKINIT)", RFC 4556,
              DOI 10.17487/RFC4556, June 2006,
              <http://www.rfc-editor.org/info/rfc4556>.

   [RFC5349] Zhu, L., Jaganathan, K., and K. Lauter, "Elliptic Curve
              Cryptography (ECC) Support for Public Key Cryptography for
              Initial Authentication in Kerberos (PKINIT)", RFC 5349,
              DOI 10.17487/RFC5349, September 2008,
              <http://www.rfc-editor.org/info/rfc5349>.

   [RFC6113] Hartman, S. and L. Zhu, "A Generalized Framework for
              Kerberos Pre-Authentication", RFC 6113,
              DOI 10.17487/RFC6113, April 2011,
              <http://www.rfc-editor.org/info/rfc6113>.

Authors' Addresses

    Michiko Short (editor)
    Microsoft Corporation
    United States of America

    Email: michikos@microsoft.com

    Seth Moore
    Microsoft Corporation
    United States of America

    Email: sethmo@microsoft.com

    Paul Miller
    Microsoft Corporation
    United States of America

    Email: paumil@microsoft.com