Kitten Working Group M. Short, Ed.
Internet-Draft S. Moore
Intended status: Standards Track P. Miller
Expires: September 7, 2015 Microsoft Corporation
March 6, 2015

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

draft-ietf-kitten-pkinit-freshness-01

Abstract

This document describes how to further extend the Public Key Cryptography for Initial Authentication in Kerberos (PKINIT) extension [RFC4556] to exchange an opaque data blob which a KDC can validate to ensure that the client is currently in possession of the private key during a PKInit AS exchange.

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 September 7, 2015.

Copyright Notice

Table of Contents

1. Introduction

The Kerberos PKINIT extension [RFC4556] defines two schemes for using asymmetric cryptography in a Kerberos preauthenticator. 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:

- 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.

- 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 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 some password-based AS exchanges [RFC4120] depend on the client sending a KRB_AS_REQ without pre-authentication to trigger the KDC to provide the Kerberos client with information needed to complete an AS exchange such as the supported encryption types and salt values (see the message flow below):

```
KDC                    Client

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

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

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

Figure 1

We can use this mechanism in PKInit for KDCs to provide data which the client returns as part of the KRB_AS_REQ to ensure that the PA_PK_AS_REQ [RFC4556] was not pregenerated.

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 RFC 2119 [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 [RFC4120] without an authenticator which includes the freshness token request to the KDC.

2. The KDC generates a KRB_ERROR as specified in Section 3.1.3 of [RFC4120] providing a freshness token.

3. The client receives the error as specified in Section 3.1.4 of [RFC4120] and includes the freshness token 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 [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 PA_AS_FRESHNESS padata type with an empty octet string as the padata-value.

2.2. Generation of KRB_ERROR Message

The KDC will respond by adding a PA_AS_FRESHNESS padata type with the freshness token as the padata-value 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 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

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 KDC_ERR_PREAUTH_FAILED with PA_AS_FRESHNESS. 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

Clients SHOULD retry in the cases when receiving a KDC_ERR_PREAUTH_FAILED KRB_ERROR message which includes a freshness token where there is a possibility that there was too much delay between the client receiving the freshness token and sending the PA_PK_AS_REQ message.

3. PreAuthentication Data Types

The following are the new PreAuthentication data types:

Padata and Data Type	Padata-type Value
PA_AS_FRESHNESS	TBD

4. Extended PKAuthenticator

The PKAuthenticator structure specified in Section 3.2.1 [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 recieved from the
            -- KDC. MUST be present if sent by KDC
   ...
}
```

5. Acknowledgements

Henry B. Hotz, Nico Williams, Sam Hartman, Tom Yu, Martin Rex, and Douglas E. Engert were key contributors to the discover of the freshness issue in PKINIT.

Greg Hudson, Nathan Ide, Benjamin Kaduk, Magnus Nystrom, Nico Williams and Tom Yu reviewed the document and provided suggestions for improvements.

6. IANA Considerations

IANA is requested to assign numbers for PA_AS_FRESHNESS listed in the Kerberos Parameters registry Pre-authentication and Typed Data as follows:

Type	Value	Reference
TBD	PA_AS_FRESHNESS	[This RFC]

7. Security Considerations

The freshness token SHOULD include signing, encrypting or sealing data from the KDC to determine authenticity and prevent tampering. Kerberos error messages are not integrity protected unless authenticated using Kerberos FAST [RFC6113]. Even if FAST is required to provide integrity protection, a different KDC would not be able to validate freshness tokens without some kind of shared database.

8. 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.

9. References

9.1. Normative References

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

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.

[RFC4556] Zhu, L. and B. Tung, "Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)", RFC 4556, June 2006.

[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, September 2008.

9.2. Informative References

[RFC6113] Hartman, S. and L. Zhu, "A Generalized Framework for Kerberos Pre-Authentication", RFC 6113, April 2011.

Authors' Addresses

Michiko Short (editor)
Microsoft Corporation
USA

Email: michikos@microsoft.com

Seth Moore
Microsoft Corporation
USA

Email: sethmo@microsoft.com

Paul Miller
Microsoft Corporation
USA

Email: paumil@microsoft.com