Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 27, 2007

G. Richards
RSA, The Security Division of EMC

June 25, 2007

OTP Kerberos
draft-richards-otp-kerberos-03

Status of this Memo

Copyright Notice

Abstract

The Kerberos protocol provides a framework authenticating a client using the exchange of pre-authentication data. This document describes the use of this framework to carry out One Time Password (OTP) authentication.

Table of Contents

1. Introduction

A One-Time Password (OTP) token may be a handheld hardware device, a hardware device connected to a personal computer through an electronic interface such as USB or a software module resident on a personal computer. All these devices generate one-time passwords that may be used to authenticate a user towards some service. This document describes a FAST [ZhHa07] factor that allows OTP values to be used in the Kerberos V5 [RFC4120] pre-authentication.

This FAST factor provides the following facilities: client-authentication, replacing-reply-key and KDC-authentication. It does not provide the strengthening-reply-key facility.

This proposal supports 4-pass and 2-pass variants. In the 4-pass system, the client sends the KDC an initial AS-REQ and the KDC response , the KDC sends with a challenge containing random nonce and details on how the OTP is to be generated as pre-auth data within a KRB-ERROR. The client generates the OTP value, the Reply Key and a Client and Server key and uses the Client Key to encrypt the nonce value. This encrypted value is then sent to the KDC along with a client generated nonce and information on the generated OTP as pre-authentication data encrypted within the armored data of a PA-FX-FAST-REQUEST contained within a second AS-REQ. The KDC then generates the same keys, verifies the pre-authentication data by decrypting the nonce and confirms knowledge of the Reply Key by sending a response to the client containing the client's nonce encrypted using the Server Key encrypted within the armored data of a PA-FX-FAST-RESPONSE contained within an AS-REP.

In the 2-pass variant, the client generates the OTP and keys as in the 4-pass system but includes the PA-FX-FAST-REQUEST in the initial AS-REQ sent to the KDC. This system can be used in cases where the client can determine in advance that OTP pre-authentication is supported by the KDC and which OTP key should be used.

Depending on OTP algorithm used, the OTP value is either used in the generation of the Reply Key or is sent to the KDC along with the encrypted nonce.

This proposal is partially based upon previous work on integrating single-use authentication mechanisms into Kerberos [HoReNeZo04] and uses the existing password-change extensions to handle PIN change as described in [RFC3244].

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].

<< This is an early draft of this document and so is liable to change significantly. >>

2. Usage Overview

2.1. Pre-Authentication

The approach uses pre-authentication data in KRB_AS_REQ, KRB_AS_REP and KRB_ERROR messages. The client begins by sending an initial KRB_AS_REQ to the KDC that may contain pre-authentication data such as the standard Kerberos password data. The KDC will then determine, in an implementation dependent fashion, whether OTP authentication is required and if it is, it will respond with a KRB_ERROR message containing a PA-OTP-CHALLENGE in the PA-DATA.

The PA-OTP-CHALLENGE will contain a KDC generated nonce and information on how the OTP should be generated by the client. The client will then generate the OTP value, its own nonce and three keys: the Reply Key, a key to encrypt KDC's nonce and a key to decrypt the KDC's reply. As described in Section 3.6, these keys will be generated from the Anchor Key and depending on the type of OTP, from the OTP value.

The encrypted KDC nonce and the client nonce along with information on how the OTP was generated are then sent to the KDC in a PA-OTP-REQUEST element encrypted within the armored-data of a PA-FX-FAST-REQUEST PA-DATA element of a second KRB_AS_REQ.

The KDC then uses this information to generate the same keys as the client, allowing it to verify the pre-authentication by decrypting the encrypted nonce sent by the client. If the validation succeeds then the KDC will confirm that the Reply Key was updated by encrypting the client's nonce and returning the encrypted value in a PA-OTP-CONFIRM element encrypted within the armored-data of a PA-FX-FAST-REPLY PA-DATA element of the KRB_AS_REP.

2.2. PIN Change

If, following successful validation of a PA-OTP-REQUEST in a KRB_AS_REQ, the KDC requires that the user changes their PIN then it will include a PA-OTP-PIN-CHANGE element in the armored data of the PA-FX-FAST-REPLY PA-DATA element of the KRB_AS_REP. This data can be used to return a new PIN to the user if the KDC has updated the PIN or to indicate to the user that they must change their PIN.

In the latter case, user PIN change shall be handled by a PIN change service supporting the ChangePasswdData in a KRB_AP_REQ as described

in [RFC3244]. If such a user PIN change is required then the KDC SHALL return a TGT in the KRB_AS_REP but it is RECOMMENDED that it only issues tickets for the PIN change service until the PIN has been changed.

2.3. Re-Synchronization

It is possible with time and event-based tokens, that the client and OTP server will loose synchronization. If, when processing a PA-OTP-REQUEST, the pre-authentication validation fails for this reason then the KDC SHALL return a KRB_ERROR message containing a PA-OTP-CHALLENGE in the PA-DATA with the "nextOTP" flag set. The setting of this flag will cause the client to re-try the authentication using the OTP for the next token "state".

3. Pre-Authentication Protocol Details

3.1. Initial Client Request

The client begins by sending an initial KRB_AS_REQ possibly containing other pre-authentication data. If the KDC determines that OTP-based pre-authentication is required and the request does not contain a PA-OTP-REQUEST then it will respond as described in Section 3.2.

Alternatively, if the client has all the necessary information, it MAY construct a PA-OTP-REQUEST as described in Section 3.3 and include it in the initial request.

3.2. KDC Challenge

If the user is required to authenticate using an OTP then the KDC SHALL respond to the initial KRB_AS_REQ with a KRB_ERROR containing:

- An error code of KDC_ERR_PREAUTH_REQUIRED
- An e-data field containing PA-DATA with a PA-OTP-CHALLENGE.

The PA-OTP-CHALLENGE SHALL contain a random nonce value to be returned encrypted in the client response. It MAY also contain information on how the OTP value is to be generated.

3.3. Client Response

The client response SHALL be sent to the KDC as a PA-OTP-REQUEST included within the enc-fast-req of a PA-FX-FAST-REQUEST encrypted under the current Anchor Key.

In order to generate its response, the client generates an OTP value. The OTP value MUST be based on the parameters in the KDC challenge if present and the response SHOULD include information on the generated OTP value.

In order to support OTP algorithms where the KDC cannot obtain the OTP value, the client MAY include the generated value in the otp-value field of the response. However, the client MUST NOT include the OTP value in the response unless it is allowed by the algorithm profile.

The client MUST generate three keys as described in Section 3.6. The generated Client Key is used by the client to encrypt data to be included in the encData of the response to allow the KDC to authenticate the user.

- If the response is being generated in response to a KDC challenge then client encrypts the value of nonce from the corresponding challenge.

- If the response is not in response to a KDC challenge then the client encrypts the current time as in the encrypted timestamp pre-authentication mechanism [RFC4120].

Finally, the client generates a random value to include in the nonce of the response. This value will then be returned encrypted by the KDC.

3.4. Verifying the pre-auth Data

The KDC validates the pre-authentication data by generating the same keys as the client as described in Section 3.6. The generated Client Key is used to decrypt the value of encData from the PA-OTP-REQUEST.

If the otp-value field is not included in the response, then the KDC SHALL use any OTP information in the PA-OTP-REQUEST to obtain the OTP value in order to generate the to generate the keys.

If the client response was sent as a result of a PA-OTP-CHALLENGE then the client authentication MUST fail if the decrypted value is not the same as the nonce value sent in the challenge. If the response was not sent as a result of a PA-OTP-CHALLENGE then the decrypted value will be a PA-ENC-TIMESTAMP and the authentication process will be the same as with standard encrypted timestamp pre-authentication [RFC4120]

3.5. Confirming the Reply Key Change

In order to support mutual authentication, the KDC SHALL respond to the clients PA-OTP-REQUEST by including in the AS-REP, the client nonce from PA-OTP-REQUEST encrypted under the Server Key.

The KDC response SHALL be sent to client as a PA_OTP_CONFIRM included within the enc-fast-rep of a PA-FX-FAST-REPLY encrypted under the current Anchor Key.

3.6. Reply Key Generation

In order to authenticate the user, the client and KDC need to generate three encryption keys:

- The Client Key to be used by the client to encrypt and by the KDC to decrypt the encData in the PA-OTP-REQUEST.
- The Server Key to be used by the KDC to encrypt and by the client to decrypt the encData value in the PA-OTP-CONFIRM.
- The Reply Key will be used in the standard manner by the KDC to encrypt data in the AS-REP.

The method used to generate the three keys will depend on the OTP algorithm.

- If the OTP value is included in the otp-value of the PA-OTP-REQUEST then all three keys SHALL be the same as the Anchor Key.
- If the OTP value is not included in the otp-value of the PA-OTP-REQUEST then the three keys SHALL be derived from the Anchor Key and the OTP value as described below.

If the OTP value is not included in the client response, then the Reply Key SHALL be generated using the KRB_FX_CF2 algorithm from [ZhHa07]

```
ClientKey = KRB_FX_CF2(K1, K2, O1, O2)
ServerKey = KRB_FX_CF2(K1, K2, O3, O4)
ReplyKey = KRB_FX_CF2(K1, K2, O5, O6)
```

The first input keys, K1, shall be the Anchor Key, AK. The second input key, K2, shall be derived from the OTP value using string-to-key or random-to-key (both defined in [RFC3961]).

- If the OTP value is binary, then K2 SHALL be derived by running the OTP value once through random-to-key.

```
K2 = random-to-key(OTP||"Krb-preAuth")
```

- If the OTP value is not binary, then K2 SHALL be derived by running the OTP value once through string-to-key.

```
K2 = string-to-key(OTP||"Krb-preAuth")
```

The salt and additional parameters for string-to-key will be as defined in section 3.1.3 of [RFC4120].

The octet string parameters, O1, O2, O3, O4, O5 and O6, shall be the ASCII string "Combine1" to "Combine6". For example, O1 and O2 have the following byte values:

```
{0x43, 0x6f, 0x6d, 0x62, 0x69, 0x6e, 0x65, 0x31}
{0x43, 0x6f, 0x6d, 0x62, 0x69, 0x6e, 0x65, 0x32}
```

4. OTP Kerberos Message Types

4.1. PA-OTP-CHALLENGE

The padata type pa-otp-challenge sent by the KDC to the client in the PA-DATA KRB_ERROR when pre-authentication using an OTP value is required. The corresponding padata-value field contains the DER encoding of a PA-OTP-CHALLENGE containing a server generated nonce and information for the client on how to generate the OTP.

```
pa-otp-challenge     << TBA >>

PA-OTP-CHALLENGE ::= SEQUENCE {
  flags             OTPFlags,
  nonce             UInt32,
  etype             INTEGER,
  otp-challenge     OCTET STRING    OPTIONAL,
  otp-length        INTEGER         OPTIONAL,
  otp-service       UTF8String      OPTIONAL,
  otp-keyID     [0] OCTET STRING    OPTIONAL,
  otp-algID     [1] INTEGER         OPTIONAL,
  ...
}

OTPFlags ::= KerberosFlags
-- nextOTP (0)
```

flags
If the "nextOTP" flag is set then the OTP calculated SHALL be based on the next token "state" rather than the current one. As an example, for a time-based token, this means the next time slot. For an event-based token, this could mean the next counter value, if counter values are used.

nonce
A KDC-supplied nonce value to be encrypted by the client in the PA-OTP-REQUEST.

etype
The encryption type to be used by the client to encrypt the nonce in the PA-OTP-REQUEST.

otp-challenge
The otp-challenge is used by the KDC to send a challenge value for use in the OTP calculation. The challenge is an optional octet string that SHOULD be uniquely generated for each request it is present in, and SHOULD be eight octets or longer when present. When the challenge is not present, the OTP will be calculated on the current token state only. The client MAY ignore a provided challenge if and only if the OTP token the client is interacting with is not capable of including a challenge in the OTP calculation. In this case, KDC policies will determine whether to accept a provided OTP value or not.

otp-length
The otp-length is used by the KDC to specify the desired length of the generated OTP.

otp-service
An identifier of the service supported by the KDC. This value can be used by the client to locate information such as the shared secret value and OTP key to use.

otp-keyID
The identifier of the OTP key to be used in the OTP calculation. If this value is not present then the client SHOULD use other values such as the otp-service and otp-algID to locate the appropriate key.

otp-algID
The identifier of the algorithm to use when generating the OTP.

<<TBD: Should a checksum be added to allow the client to verify the challenge?>>

4.2. PA-OTP-REQUEST

The padata-type pa-otp-response sent by the client to the KDC in the KrbFastReq padata of a PA-FX-FAST-REQUEST included in the PA-DATA of an AS-REQ. The corresponding padata-value field contains the DER encoding of a PA-OTP-REQUEST.

The message contains pre-authentication data encrypted by the client using the generated Client Key and information on how the OTP was generated. It may also,optionally, contain the generated OTP value.

```
        pa-otp-response     << TBA >>

        PA-OTP-REQUEST ::= SEQUENCE {
          flags             OTPFlags,
          nonce             UInt32,
          encData           EncryptedData,
                            -- PA-OTP-ENC-REQUEST or PA-ENC-TIMESTAMP
                            -- Key usage of << TBD >>
          otp-value         OCTET STRING    OPTIONAL,
          otp-challenge [0] OCTET STRING    OPTIONAL,
          otp-time          KerberosTime    OPTIONAL,
          otp-counter   [1] OCTET STRING    OPTIONAL,
          otp-format        OTPFormat       OPTIONAL,
          otp-keyID     [2] OCTET STRING    OPTIONAL,
          otp-algID     [3] INTEGER         OPTIONAL,
          ...
        }


          PA-OTP-ENC-REQUEST ::= SEQUENCE {
            nonce     OCTET STRING,
            ...
          }


          OTPFormat ::= INTEGER {
            decimal(0),
            hexadecimal(1),
            alphanumeric(2),
            binary(3)
          }
```

flags

If the "nextOTP" flag is set then the OTP was calculated based on the next token "state" rather than the current one. This flag MUST be set if and only if it was set in a corresponding PA-OTP-CHALLENGE.

nonce
A random nonce value generated by the client.

encData
If the PA-OTP-REQUEST is sent as a result of a PA-OTP_CHALLENGE then this MUST contain the nonce from the challenge encrypted under the client key. If no challenge was received then this MUST contain a PA-ENC-TIMESTAMP encrypted under the client key.

otp-value
The generated OTP value. This value MUST NOT be present unless allowed by the OTP algorithm profile.

otp-challenge
Value used by the client in the OTP calculation. It MUST be sent to the KDC if and only if the value would otherwise be unknown to the KDC. For example, the token or client modified or generated challenge.

otp-time
Value used by the client to send the time used in the OTP calculation.

otp-counter
The counter value used in the OTP calculation. Use of this element is OPTIONAL but it MAY be used by a client to simplify the OTP calculations of the KDC to contain the counter value as reported by the OTP token.

otp-format
The format of the generated OTP.

otp-keyID
The identifier of the OTP key used.

otp-algID
The identifier of the algorithm to used to generate the OTP.

4.3. PA-OTP-CONFIRM

This pre-authentication type is returned by the KDC in the enc-fast-rep of a PA-FX_FAST-REPLY in the KRB_AS_REP of the KDC. It is used to return the client's nonce encrypted under the new Server Key in order to confirm that the KDC has knowledge of this key.

```
pa-otp-confirm     << TBA >>

PA-OTP-CONFIRM ::= SEQUENCE {
  encData        EncryptedData,
                       -- PA-OTP-ENC-CONFIRM
                       -- Key usage of << TBD >>
  ...
}


PA-OTP-ENC-CONFIRM ::= SEQUENCE {
  nonce     OCTET STRING,
  ...
}
```

encData
The value of nonce from the corresponding PA-OTP-REQUEST encrypted under the current Server Key.

4.4. PA-OTP-PIN-CHANGE

This pre-authentication type returned by the KDC in the enc-fast-rep of a PA-FX-FAST-REPLY in the KRB_AS_REP if the user must change their PIN or if the user's PIN has been changed.

```
PA-OTP-PIN-CHANGE ::= SEQUENCE {
  flags         PinFlags,
  pin           UTF8String OPTIONAL,
  minLength     INTEGER    OPTIONAL,
  maxLength [1] INTEGER    OPTIONAL,
  ...
}

PinFlags ::= KerberosFlags
  -- systemSetPin (0)
```

If the "systemSetPin" flag is set then the user's PIN has been changed and the new PIN value is contained in the pin field. The pin field MUST therefore be present.

If the "systemSetPin" flag is not set then the user's PIN has not been changed by the server but it MUST instead be changed by the user using the PIN change service. Restrictions on the size of the PIN MAY be given by the minLength and maxLength fields. If the pin field is present then it contains a PIN value that MAY be used by the user when changing the PIN. The KDC MAY only issue tickets for the PIN change service until the PIN has been changed.

5. IANA Considerations

A registry may be required for the otp-AlgID values as introduced in Section 4.1. No other IANA actions are anticipated.

6. Security Considerations

6.1. Active attacks

6.1.1. Man-in-the-Middle

In the system described in this document, the OTP pre-authentication protocol is tunneled within the FAST Armor channel provided by the pre-authentication framework. As described in [AsNiNy02], tunneled protocols are potentially vulnerable to man-in-the-middle attacks if the outer tunnel is compromised and it is generally considered good practice in such cases to bind the inner encryption to the outer tunnel.

Even though no such attacks are known at this point, the proposed system uses the outer, Anchor, Key in the derivation of the inner Client and Server keys and so achieve crypto-binding to the outer channel.

6.1.2. Reflection

The 4-pass system described above is a challenge-response protocol and such protocols are potentially vulnerable to reflection attacks. No such attacks are known at this point but to help mitigate against such attacks, the system uses different keys to encrypt the client and server nonces.

6.1.3. Replay

The 2-pass version of the protocol does not involve a server nonce and so the client instead encrypts a timestamp. To reduce the chance of replay attacks, the KDC must check that the client time used in such a request is later than that used in previous requests.

6.1.4. Passive attacks

<< TBD >>

6.2. FAST Facilities

The secret used to generate the OTP is known only to the client and the KDC and so successful decryption of the encrypted nonce by the

KDC authenticates the user. Similarly, successful decryption of the encrypted nonce by the client proves that the expected KDC replied. The Reply Key is replaced by a key generated from the OTP and Armor key. This FAST factor therefore provides the following facilities: client-authentication, replacing-reply-key and KDC-authentication.

7. References

7.1. Normative References

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

[RFC3244] Swift, M., Trostle, J., and J. Brezak, "Microsoft Windows 2000 Kerberos Change Password and Set Password Protocols", RFC 3244, February 2002.

[RFC3961] Raeburn, K., "Encryption and Checksum Specifications for Kerberos 5", RFC 3961, February 2005.

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

[ZhHa07] Znu, L. and S. Hartman, "A generalized Framework for Kerberos Pre-Authentication", draft-ietf-kerb-wg-preauth-framework-05 (work in progress), March 2007.

7.2. Informative References

[AsNiNy02] Asokan, N., Niemi, V., and K. Nyberg, "Man-in-the-Middle in Tunneled Authentication Protocols", Cryptology ePrint Archive Report 2002/163, November 2002.

[HoReNeZo04] Horstein, K., Renard, K., Neuman, C., and G. Zorn, "Integrating Single-use Authentication Mechanisms with Kerberos", draft-ietf-krb-wg-kerberos-sam-03 (work in progress), July 2004.

Appendix A. ASN.1 Module

```
OTPKerberos
DEFINITIONS IMPLICIT TAGS ::=
```

```
BEGIN

  IMPORTS
      KerberosTime, KerberosFlags, EncryptionKey, UInt32,
      Int32, EncryptedData
      FROM KerberosV5Spec2 {iso(1) identified-organization(3)
                            dod(6) internet(1) security(5) kerberosV5(2)
                            modules(4) krb5spec2(2) };
                             -- as defined in RFC 4120.

    PA-OTP-CHALLENGE ::= SEQUENCE {
      flags             OTPFlags,
      nonce             UInt32,
      etype             INTEGER,
      otp-challenge     OCTET STRING    OPTIONAL,
      otp-length        INTEGER         OPTIONAL,
      otp-service       UTF8String      OPTIONAL,
      otp-keyID     [0] OCTET STRING    OPTIONAL,
      otp-algID     [1] INTEGER         OPTIONAL,
      ...
    }

    OTPFlags ::= KerberosFlags
    -- nextOTP (0)

    PA-OTP-REQUEST ::= SEQUENCE {
      flags             OTPFlags,
      nonce             UInt32,
      encData           EncryptedData,
                        -- PA-OTP-ENC-REQUEST or PA-ENC-TIMESTAMP
                        -- Key usage of << TBD >>
      otp-value         OCTET STRING    OPTIONAL,
      otp-challenge [0] OCTET STRING    OPTIONAL,
      otp-time          KerberosTime    OPTIONAL,
      otp-counter   [1] OCTET STRING    OPTIONAL,
      otp-format        OTPFormat       OPTIONAL,
      otp-keyID     [2] OCTET STRING    OPTIONAL,
      otp-algID     [3] INTEGER         OPTIONAL,
            ...
    }

    PA-OTP-ENC-REQUEST ::= SEQUENCE {
      nonce     OCTET STRING,
      ...
    }

    OTPFormat ::= INTEGER {
      decimal(0),
```

```
      hexadecimal(1),
      alphanumeric(2),
      binary(3)
    }

    PA-OTP-CONFIRM ::= SEQUENCE {
      encData        EncryptedData,
                     -- PA-OTP-ENC-CONFIRM
                     -- Key usage of << TBD >>
      ...
    }

    PA-OTP-ENC-CONFIRM ::= SEQUENCE {
      nonce     OCTET STRING,
      ...
    }

    PA-OTP-PIN-CHANGE ::= SEQUENCE {
      flags         PinFlags,
      pin           UTF8String OPTIONAL,
      minLength     INTEGER    OPTIONAL,
      maxLength [0] INTEGER    OPTIONAL,
      ...
    }

    PinFlags ::= KerberosFlags
    -- systemSetPin (0)

END
```

Author's Address

Gareth Richards
RSA, The Security Division of EMC
RSA House
Western Road
Bracknell, Berkshire RG12 1RT
UK

Email: grichards@rsa.com

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).