Network Working Group
Internet-Draft
Expires: August 3, 2003

S. Josefsson
February 2, 2003

A Kerberos 5 SASL Mechanism
draft-josefsson-sasl-kerberos5-01

Status of this Memo

Copyright Notice

Abstract

This document specifies a Simple Authentication and Security Layer (SASL) [3] mechanism for Kerberos 5 Client/Server Authentication [1], with optional initial Authentication Service (AS) and/or Ticket-Granting-Service (TGS) exchanges.

Table of Contents

1. Introduction

Kerberos 5 provides client and optional server authentication, usually employing symmetric encryption and a trusted (symmetric) key distribution center. Specifying Kerberos 5 authentication for each network protocol where there is a need to use Kerberos 5 is a tedious task. However, as many protocols already specify support for the SASL framework, by specifying a Kerberos 5 SASL mechanism, support for Kerberos 5 in many protocols is accomplished. Even for protocols that do not support SASL, specifying SASL support (and thereby implicitly Kerberos 5) is often advantageous over specifying Kerberos 5 support directly. The advantages include better flexibility if or when new Kerberos versions are released, and perhaps more commonly, when circumstances demand that other authentication methods (supported by SASL) should be used.

It should be mentioned that Kerberos 5 authentication via SASL is already possible, by using the Generic Security Service Application Program Interface [6] framework. However, GSSAPI adds some amount of overhead, both in terms of code complexity, code size and additional network round trips. More importantly, GSSAPI do not support the authentication steps (AS and TGS). These are some of the motivation behind this "slimmer" Kerberos 5 SASL mechanism.

2. Document Changes

Modification since -00:

* The way data is encoded in the AP-REQ.Authenticator.authorization-data field is corrected and elaborated.

* The incorrect sentence about including application data in the AP-REP is removed.

* The "Theory of operation" section now includes three modes; Infrastructure, Proxied Infrastructure, and Non-Infrastructure modes.

* The example section now contains a complete dump from an implementation.

3. Kerberos Version 5 Mechanism

The mechanism name associated with Kerberos version 5 is "KERBEROS_V5". The exchange consists of one initial server packet (containing some parameters and a challenge, described below), and

then an unfixed number of messages containing Kerberos 5 packets, with the last exchange being an AP-REQ, and optional AP-REP, for the desired SASL service on a format described below.

The normal packet exchange, after the required initial server packet, are one optional AS-REQ and AS-REP exchange, one optional TGS-REQ and TGS-REP exchange and then the AP-REQ packet and optional AP-REP reply. The only steps that are required by this SASL mechanism is the initial server packet and the final AP-REQ and optional AP-REP exchange. The AP-REP is sent when and only when mutual authentication is required. The AP-REQ is for the SASL service that is requested. The AP-REQ must contain authenticated application data on a format described below. The AS and TGS exchanges is usually used by clients to acquire the proper tickets required for the AP phase. It is not expected that any other Kerberos 5 packets will be exchanged, but this mechanism do not disallow such packets in order to make it possible to use this SASL mechanism with future Kerberos extensions.

As discussed above, the client is allowed to send any valid Kerberos 5 message and the server should handle any message, subject to local policy restrictions. If the server do not understand the meaning of a packet or do not wish to respond to it (e.g., it cannot proxy a TGS-REQ), it SHOULD respond with a empty response and await further packets. Reasons for aborting the authentication phase instead of sending an empty response includes if the number of received packets exceeds a pre-defined limit. AS-REQ and TGS-REQ packets destined for non-local Kerberos Key Distribution Centers (KDCs) is proxied to the correct server by the SASL server. No provisions are made in the protocol to allow the client to specify the addresses of the KDCs, instead the SASL server is required to discover this information (usually by static configuration or by using DNS). It is legitimate for the SASL server to abort the authentication phase with an error saying that the indicated realm was not found or is restricted by policy (i.e., a policy that only permits local KDC requests is permitted).

Since it is expected that clients may not yet have IP addresses when they invoke this SASL mechanism (invoking this mechanism may be one step in acquiring an IP address), clients commonly leave the address fields in the AS-REQ empty.

The initial server packet should contain one octet containing a bit mask of supported security layers, four octets indicating the maximum cipher-text buffer size the server is able to receive (or 0 if no security layers are supported) in network byte order, and then 16 octets containing random data (see [4] on how random data might be generated).

The last exchange must be an AP-REQ for the desired SASL service, optionally followed by an AP-REP. The SASL service is translated into a Kerberos principal and realm as follows: The first element of the principal is the service name specified in the protocol that uses SASL. The second element is the address of the SASL server, usually expressed as a hostname. The SASL realm should be the Kerberos realm of the server. The checksum value in the "cksum" field in the Authenticator in the AP-REQ is computed on a string where the first octet indicate the desired security layer requested by the client (a bitmask with one bit set, which must also be set in the security layer bitmask offered by the server), the next four octets indicate the maximum cipher-text buffer size the client is able to receive in network byte order (or 0 if a security layer is not indicated by the first octet), followed by the entire initial server packet, in turn followed by the desired authorization identity (which can be empty to indicate that the authorization identity should be the same as the authentication identity in the Kerberos ticket stored in the AP-REQ). This same string is also to be included in the authorization-data field of the Authenticator, with an ad-type of -1.

Upon decrypting and verifying the ticket and authenticator (i.e., standard AP-REQ processing), the server extracts the authorization-data value from the Authenticator and checks that the checksum in the authenticator is correct. It then proceeds to check that the server security layer bit mask, server maximum cipher-text buffer size, and the random data equals the data provided in the initial server challenge. The server verify that the client selected a security layer that was offered, and that the client maximum buffer is 0 if no security layer was chosen. The server must also verify that the principal identified in the Kerberos ticket is authorized to connect to the service as the authorization identity specified by the client (or, if absent, the username denoted by the ticket principal). Unless the client requested mutual authentication, the authentication process is complete.

If the client requested mutual authentication, the server constructs an AP-REP according to Kerberos 5.

The security layers and their corresponding bit-masks are as follows:

```
Bit 0 No security layer
Bit 1 Integrity (KRB-SAFE) protection
Bit 2 Privacy (KRB-PRIV) protection
Bit 3 Mutual authentication is required (AP option MUTUAL-
      REQUIRED must also be present).
```

Other bit-masks may be defined in the future; bits which are not understood must be negotiated off.

4. Theory Of Operation

This section describes, for illustration only, three common scenarios where this mechanism can be used.

4.1 Infrastructure Mode

Normally a SASL server is an application server such as a mail system. The server is configured to belong to at least one Kerberos 5 realm, and the server shares a symmetric key with the Kerberos 5 Key Distribution Center in that realm. The server cannot perform the initial Kerberos AS and TGS operation itself, but a KDC is needed for that operation. Once the user acquired credentials the server is able to carry out the AP-REQ/AP-REP phase using its symmetric key. The steps are as follows:

0) Server sends initial token.

* Client acquires a ticket for the server using an out-of-band request to the KDC. Client generates the AP-REQ using the ticket for the server.

1) Client sends AP-REQ to the server.

* Server parses AP-REQ, and if required the AP-REP is generated.

2) [Optional] Server sends AP-REP to the client.

* [Optional] Client parses AP-REP.

As can be seen, round-trip wise this is optimal, possibly bar the initial token, although in IMAP it does not cause an additional round-trip, and other protocols may be similar.

4.2 Proxied Infrastructure Mode

If the client for some reason cannot carry out the communication with the KDC itself, or for some other reason the server is in a better position than the client to communicate with the KDC, the server can proxy that part of the exchange via the server using the SASL framework. The server effectively acts as a proxy. Note that the packets that are sent are identical to those in the first example, they are only routed differently. The steps are as follows:

0) Server sends initial token.

* Client constructs AS-REQ using username and realm supplied by user, in order to acquire a ticket granting ticket.

1) Client sends AS-REQ to server.

* Server finds address of KDC and forwards the AS-REQ to and waits for the AS-REP response from the KDC.

2) Server sends AS-REP to client.

* Client parses AS-REP and constructs a TGS-REQ using the ticket granting ticket encryption key, in order to acquire a ticket for the server.

3) Client sends TGS-REQ to server.

* Server finds address of KDC and forwards the TGS-REQ to and waits for the TGS-REP response from the KDC.

4) Server sends TGS-REP to client.

* Client parses TGS-REP and generates the AP-REQ using the session encryption key.

5) Client sends AP-REQ to server.

* Server parses AP-REQ and if required the AP-REP is generated.

6) [Optional] Server sends AP-REP.

* [Optional] Client parses AP-REP.

If efficiency as a concern, and the client have no other use of a ticket granting ticket for the realm, step 3 and 4 can be skipped by asking for a ticket for the server directly in the AS-REQ.

Note that the client in subsequent connections may try to re-use the ticket negotiated if it is still valid.

4.3 Non-Infrastructure Mode

Kerberos 5 is usually a distributed security system, but we wish to point out that this Kerberos 5 SASL mechanism may be used in a standalone SASL server to provide the security advantages that the Kerberos 5 Authentication Service (AS) provides over other methods.

Specifically, the SASL server may use a legacy password database such as a CRAM-MD5 clear text password file to generate Kerberos 5 principals "on the fly". Authentication may thus proceed as follows:

0) Server sends initial token.

* Client constructs AS-REQ using username supplied by user, in order to acquire a ticket for the server directly. The realm can be predetermined by administrators, or simply the hostname of the server.

1) Client sends AS-REQ to server.

* Server parses AS-REQ and generates AS-REP based on password in database. The AS-REQ embeds a ticket for the server.

2) Server sends AS-REP to client.

* Client parses AS-REP and extracts the ticket and generates an AP-REQ using the session encryption key.

3) Client sends AP-REQ to server.

* Server parses AP-REQ and if required, generates the AP-REP.

4) [Optional] Server sends AP-REP to client.

* [Optional] Client parses AP-REP.

This may be extended further, i.e. by using the password and Kerberos 5 pre-authentication in step 1.

Note that the client in subsequent connections may try to re-use the ticket negotiated if it is still valid.

5. Example

The following is one Kerberos version 5 login scenario for the IMAP4 protocol, in the non-infrastructure mode. Note that the line breaks are for editorial clarity.

```
S: * OK IMAP4rev1 server
C: . AUTHENTICATE KERBEROS_V5
S: + CQAAAADp6+ONC2vcprRbmH2J95Gh
C: an4wfKEDAgEFogMCAQqkcDBuoAcDBQAAAAAAoRAwDqADAgEAoQcwBRsDamFzog
   sbCWxvY2FsaG9zdKMcMBqgAwIBAKETMBEbBGltYXAbCWxvY2FsaG9zdKURGA8y
   MDAzMDIwMjE2NDE0M1qnBgIEVAbYn6gLMAkCARECARACAQM=
S: + a4IBzDCCAcigAwIBBaEDAgELowsbCWxvY2FsaG9zdKQQMA6gAwIBAKEHMAUb
   A2phc6WB5GGB4TCB3qADAgEFoQsbCWxvY2FsaG9zdKIcMBqgAwIBAaETMBEbBG
   ltYXAbCWxvY2FsaG9zdKOBqzCBqKADAgESooGgBIGdeBv2/NG1EgTMMMcHaVY3
   f2w6y+bA56cVP8Toh+A3XFvTw8JFqAJVFGDm3MBrrSFOYcN8/8WY8T1cm0jq68
   TcsiMh8y9KbWyeLJZedCVLLIfP1JgsSbBkZ7NLBFYCEEKvoGz2lMAuyJSh4+zT
   L/NbcoIJq2ynCS965JKWWXl4rcBZPKBn5YUoU71dRK4/+HFrBoHejr6UHwVKd/
   y0TaaBtTCBsqADAgERooGqBIGnYP7dngXFL2/hUWEs5PxGwlmvpWzGHWyh2QJ7
   52eFj1tUpU3qT1NGgfVq2BVVWGDSVTO1vgDrkKCSDQwzkrqfwoZh4t6tt5tAPn
   MCx2VDGyOu4Uv4PUsw4+uEevqkQRczpCsZT+y7pX7CxWHtytT3vLXNA6sANGnu
   O7v7gTO+MGxzNvhVgMlujT2dkVgvCviVgJNuVef1VLVJWYM/zc4tuPaPaWToZJ
   c=
C: boIBvjCCAbqgAwIBBaEDAgEOogcDBQAEAAAAo4HkYYHhMIHeoAMCAQWhCxsJbG
   9jYWxob3N0ohwwGqADAgEBoRMwERsEaW1hcBsJbG9jYWxob3N0o4GrMIGooAMC
   ARKigaAEgZ14G/b80bUSBMwwxwdpVjd/bDrL5sDnpxU/xOiH4DdcW9PDwkWoAl
   UUYObcwGutIU5hw3z/xZjxPVybSOrrxNyyIyHzL0ptbJ4sll50JUssh8/UmCxJ
   sGRns0sEVgIQQq+gbPaUwC7IlKHj7NMv81tyggmrbKcJL3rkkpZZeXitwFk8oG
   flhShTvV1Erj/4cWsGgd6OvpQfBUp3/LRNpIG9MIG6oAMCARGhAwIBAaKBrQSB
   qjE+doGGFMaz8g+nKl45qG5BPxzql0jXI5YMS+JqDeNBJIKasB0v9wMzXP9t8L
   62PLsanqpow5bxAUtl/Dc8hqvc0cB+cC1P8RTgb0upMqzxTristf7goWRhQgTJ
   OOwKJp/ZftZOkSdTHBQZL8StYuYe/6RkKkgnkUMK10VSec/YamG+5s37GvoRPG
   Hu126PTyjXs3EziFqf6HT9Da/NJnDClFL8+nnlVFVt
S: + b2IwYKADAgEFoQMCAQ+iVDBSoAMCARGhAwIBAKJGBESTDM1z2PF5cUYOBmOW
   IouXfHWtQzYzj1JFsJMV/CHMTmBrJavImHjR24f9WyCNOvmJMAWeHHOV9Jtpj6
   rFt/ytas4U0g==
C:
S: . OK AUTHENTICATE KERBEROS_V5 authentication successful
```

The service requested is "imap/localhost" in the realm "localhost". The password used was "foo", yielding an aes256-cts-hmac-sha1-96 encryption key of 0x6aefbaf05423cbc0fb44a41cc221783d7f52b764cca41fe2a05ad6d3bc7a67ea.

The first packet specify that mutual authentication and no integrity or privacy layer (hence a zero maximum buffer size) and some random data.

The second packet contains the AS-REQ, expanded as follows:

```
   name:KDC-REQ  type:SEQUENCE
     name:pvno  type:INTEGER  value:0x05
     name:msg-type  type:INTEGER  value:0x0a
     name:req-body  type:SEQUENCE
       name:kdc-options  type:BIT_STR  value(32):00000000
       name:cname  type:SEQUENCE
         name:name-type  type:INTEGER  value:0x00
         name:name-string  type:SEQ_OF
           name:NULL  type:GENERALSTRING
           name:?1  type:GENERALSTRING  value:6a6173
       name:realm  type:GENERALSTRING  value:6c6f63616c686f7374
       name:sname  type:SEQUENCE
         name:name-type  type:INTEGER  value:0x00
         name:name-string  type:SEQ_OF
           name:NULL  type:GENERALSTRING
           name:?1  type:GENERALSTRING  value:696d6170
           name:?2  type:GENERALSTRING  value:6c6f63616c686f7374
       name:till  type:TIME  value:20030202164143Z
       name:nonce  type:INTEGER  value:0x5406d89f
       name:etype  type:SEQ_OF
         name:NULL  type:INTEGER
         name:?1  type:INTEGER  value:0x11
         name:?2  type:INTEGER  value:0x10
         name:?3  type:INTEGER  value:0x03
   -----BEGIN SHISHI KDC-REQ-----
   an4wfKEDAgEFogMCAQqkcDBuoAcDBQAAAAAoRAwDqADAgEAoQcwBRsDamFzogsb
   CWxvY2FsaG9zdKMcMBqgAwIBAKETMBEbBGltYXAbCWxvY2FsaG9zdKURGA8yMDAz
   MDIwMjE2NDE0M1qnBgIEVAbYn6gLMAkCARECARACAQM=
   -----END SHISHI KDC-REQ-----

   The third packet contains the AS-REP, expanded as follows:

   name:KDC-REP  type:SEQUENCE
     name:pvno  type:INTEGER  value:0x05
     name:msg-type  type:INTEGER  value:0x0b
     name:crealm  type:GENERALSTRING  value:6c6f63616c686f7374
     name:cname  type:SEQUENCE
       name:name-type  type:INTEGER  value:0x00
       name:name-string  type:SEQ_OF
         name:NULL  type:GENERALSTRING
         name:?1  type:GENERALSTRING  value:6a6173
     name:ticket  type:SEQUENCE
       name:tkt-vno  type:INTEGER  value:0x05
       name:realm  type:GENERALSTRING  value:6c6f63616c686f7374
       name:sname  type:SEQUENCE
         name:name-type  type:INTEGER  value:0x01
         name:name-string  type:SEQ_OF
           name:NULL  type:GENERALSTRING
```

```
          name:?1  type:GENERALSTRING  value:696d6170
          name:?2  type:GENERALSTRING  value:6c6f63616c686f7374
      name:enc-part  type:SEQUENCE
        name:etype  type:INTEGER  value:0x12
        name:cipher  type:OCT_STR  value:781bf6fcd1b51204cc30c7076956377
        f6c3acbe6c0e7a7153fc4e887e0375c5bd3c3c245a802551460e6dcc06bad214
        e61c37cffc598f13d5c9b48eaebc4dcb22321f32f4a6d6c9e2c965e74254b2c8
        7cfd4982c49b06467b34b0456021042afa06cf694c02ec894a1e3ecd32ff35b7
        28209ab6ca7092f7ae49296597978adc0593ca067e5852853bd5d44ae3ff8716
        b0681de8ebe941f054a77fcb44d
    name:enc-part  type:SEQUENCE
      name:etype  type:INTEGER  value:0x11
      name:cipher  type:OCT_STR  value:60fedd9e05c52f6fe151612ce4fc46c25
      9afa56cc61d6ca1d9027be767858f5b54a54dea4f534681f56ad815555860d2553
      3b5be00eb90a0920d0c3392ba9fc28661e2deadb79b403e7302c765431b23aee14
      bf83d4b30e3eb847afaa4411733a42b194fecbba57ec2c561edcad4f7bcb5cd03a
      b003469ee3bbbfb8133be306c7336f85580c96e8d3d9d91582f0af89580936e55e
      7f554b54959833fcdce2db8f68f6964e86497
  -----BEGIN SHISHI KDC-REP-----
  a4IBzDCCAcigAwIBBaEDAgELowsbCWxvY2FsaG9zdKQQMA6gAwIBAKEHMAUbA2ph
  c6WB5GGB4TCB3qADAgEFoQsbCWxvY2FsaG9zdKIcMBqgAwIBAaETMBEbBGltYXAb
  CWxvY2FsaG9zdKOBqzCBqKADAgESooGgBIGdeBv2/NG1EgTMMMcHaVY3f2w6y+bA
  56cVP8Toh+A3XFvTw8JFqAJVFGDm3MBrrSFOYcN8/8WY8T1cm0jq68TcsiMh8y9K
  bWyeLJZedCVLLIfP1JgsSbBkZ7NLBFYCEEKvoGz2lMAuyJSh4+zTL/NbcoIJq2yn
  CS965JKWWXl4rcBZPKBn5YUoU71dRK4/+HFrBoHejr6UHwVKd/y0TaaBtTCBsqAD
  AgERooGqBIGnYP7dngXFL2/hUWEs5PxGwlmvpWzGHWyh2QJ752eFj1tUpU3qT1NG
  gfVq2BVVWGDSVTO1vgDrkKCSDQwzkrqfwoZh4t6tt5tAPnMCx2VDGyOu4Uv4PUsw
  4+uEevqkQRczpCsZT+y7pX7CxWHtytT3vLXNA6sANGnuO7v7gTO+MGxzNvhVgMlu
  jT2dkVgvCviVgJNuVef1VLVJWYM/zc4tuPaPaWToZJc=
  -----END SHISHI KDC-REP-----
```

After extracting the AS-REP, the EncASRepPart and Ticket are as follows:

```
   name:EncKDCRepPart  type:SEQUENCE
     name:key  type:SEQUENCE
       name:keytype  type:INTEGER  value:0x11
       name:keyvalue  type:OCT_STR  value:517fe065071c845c425b5b18c4236618
     name:last-req  type:SEQ_OF
       name:NULL  type:SEQUENCE
         name:lr-type  type:INTEGER
         name:lr-value  type:TIME
     name:nonce  type:INTEGER  value:0x5406d89f
     name:flags  type:BIT_STR  value(3):20
     name:authtime  type:TIME  value:20030202162503Z
     name:endtime  type:TIME  value:20030202164143Z
     name:srealm  type:GENERALSTRING  value:6c6f63616c686f7374
     name:sname  type:SEQUENCE
       name:name-type  type:INTEGER  value:0x01
       name:name-string  type:SEQ_OF
         name:NULL  type:GENERALSTRING
         name:?1  type:GENERALSTRING  value:696d6170
         name:?2  type:GENERALSTRING  value:6c6f63616c686f7374
   -----BEGIN SHISHI EncKDCRepPart-----
   MIGAoBswGaADAgERoRIEEFF/4GUHHIRcQltbGMQjZhihAjAAogYCBFQG2J+kBAMC
   BSClERgPMjAwMzAyMDIxNjI1MDNapxEYDzIwMDMwMjAyMTY0MTQzWqkLGwlsb2Nh
   bGhvc3SqHDAaoAMCAQGhEzARGwRpbWFwGwlsb2NhbGhvc3Q=
   -----END SHISHI EncKDCRepPart-----
```

```
   name:Ticket  type:SEQUENCE
     name:tkt-vno  type:INTEGER  value:0x05
     name:realm  type:GENERALSTRING  value:6c6f63616c686f7374
     name:sname  type:SEQUENCE
       name:name-type  type:INTEGER  value:0x01
       name:name-string  type:SEQ_OF
         name:NULL  type:GENERALSTRING
         name:?1  type:GENERALSTRING  value:696d6170
         name:?2  type:GENERALSTRING  value:6c6f63616c686f7374
     name:enc-part  type:SEQUENCE
       name:etype  type:INTEGER  value:0x12
       name:cipher  type:OCT_STR  value:781bf6fcd1b51204cc30c7076956377f6
       c3acbe6c0e7a7153fc4e887e0375c5bd3c3c245a802551460e6dcc06bad214e61c
       37cffc598f13d5c9b48eaebc4dcb22321f32f4a6d6c9e2c965e74254b2c87cfd49
       82c49b06467b34b0456021042afa06cf694c02ec894a1e3ecd32ff35b728209ab6
       ca7092f7ae49296597978adc0593ca067e5852853bd5d44ae3ff8716b0681de8eb
       e941f054a77fcb44d
   -----BEGIN SHISHI Ticket-----
   YYHhMIHeoAMCAQWhCxsJbG9jYWxob3N0ohwwGqADAgEBoRMwERsEaW1hcBsJbG9j
   YWxob3N0o4GrMIGooAMCARKigaAEgZ14G/b80bUSBMwwxwdpVjd/bDrL5sDnpxU/
   xOiH4DdcW9PDwkWoAlUUYObcwGutIU5hw3z/xZjxPVybSOrrxNyyIyHzL0ptbJ4s
   ll50JUssh8/UmCxJsGRns0sEVgIQQq+gbPaUwC7IlKHj7NMv81tyggmrbKcJL3rk
   kpZZeXitwFk8oGflhShTvV1Erj/4cWsGgd6OvpQfBUp3/LRN
   -----END SHISHI Ticket-----

   The third packet contains the AP-REQ, expanded as follows:

   name:AP-REQ  type:SEQUENCE
     name:pvno  type:INTEGER  value:0x05
     name:msg-type  type:INTEGER  value:0x0e
     name:ap-options  type:BIT_STR  value(32):04000000
     name:ticket  type:SEQUENCE
       name:tkt-vno  type:INTEGER  value:0x05
       name:realm  type:GENERALSTRING  value:6c6f63616c686f7374
       name:sname  type:SEQUENCE
         name:name-type  type:INTEGER  value:0x01
         name:name-string  type:SEQ_OF
           name:NULL  type:GENERALSTRING
           name:?1  type:GENERALSTRING  value:696d6170
           name:?2  type:GENERALSTRING  value:6c6f63616c686f7374
       name:enc-part  type:SEQUENCE
         name:etype  type:INTEGER  value:0x12
         name:cipher  type:OCT_STR  value:781bf6fcd1b51204cc30c7076956377
         f6c3acbe6c0e7a7153fc4e887e0375c5bd3c3c245a802551460e6dcc06bad214
         e61c37cffc598f13d5c9b48eaebc4dcb22321f32f4a6d6c9e2c965e74254b2c8
         7cfd4982c49b06467b34b0456021042afa06cf694c02ec894a1e3ecd32ff35b7
         28209ab6ca7092f7ae49296597978adc0593ca067e5852853bd5d44ae3ff8716
         b0681de8ebe941f054a77fcb44d
```

```
    name:authenticator  type:SEQUENCE
      name:etype  type:INTEGER  value:0x11
      name:kvno  type:INTEGER  value:0x01
      name:cipher  type:OCT_STR  value:313e76818614c6b3f20fa72a5e39a86e4
      13f1cea9748d723960c4be26a0de34124829ab01d2ff703335cff6df0beb63cbb1
      a9eaa68c396f1014b65fc373c86abdcd1c07e702d4ff114e06f4ba932acf14eb8a
      cb5fee0a164614204c938ec0a269fd97ed64e9127531c14192fc4ad62e61effa46
      42a482791430ad7455279cfd86a61bee6cdfb1afa113c61eed76e8f4f28d7b3713
      3885a9fe874fd0dafcd2670c29452fcfa79e554556d
  -----BEGIN SHISHI AP-REQ-----
  boIBvjCCAbqgAwIBBaEDAgEOogcDBQAEAAAAo4HkYYHhMIHeoAMCAQWhCxsJbG9j
  YWxob3N0ohwwGqADAgEBoRMwERsEaW1hcBsJbG9jYWxob3N0o4GrMIGooAMCARKi
  gaAEgZ14G/b80bUSBMwwxwdpVjd/bDrL5sDnpxU/xOiH4DdcW9PDwkWoAlUUYObc
  wGutIU5hw3z/xZjxPVybSOrrxNyyIyHzL0ptbJ4sll50JUssh8/UmCxJsGRns0sE
  VgIQQq+gbPaUwC7IlKHj7NMv81tyggmrbKcJL3rkkpZZeXitwFk8oGflhShTvV1E
  rj/4cWsGgd6OvpQfBUp3/LRNpIG9MIG6oAMCARGhAwIBAaKBrQSBqjE+doGGFMaz
  8g+nKl45qG5BPxzql0jXI5YMS+JqDeNBJIKasB0v9wMzXP9t8L62PLsanqpow5bx
  AUtl/Dc8hqvc0cB+cC1P8RTgb0upMqzxTristf7goWRhQgTJOOwKJp/ZftZOkSdT
  HBQZL8StYuYe/6RkKkgnkUMK10VSec/YamG+5s37GvoRPGHu126PTyjXs3EziFqf
  6HT9Da/NJnDClFL8+nnlVFVt
  -----END SHISHI AP-REQ-----
```

After extracting the AP-REP, the Authenticator is as follows:

```
name:Authenticator  type:SEQUENCE
  name:authenticator-vno  type:INTEGER  value:0x05
  name:crealm  type:GENERALSTRING  value:6c6f63616c686f7374
  name:cname  type:SEQUENCE
    name:name-type  type:INTEGER  value:0x01
    name:name-string  type:SEQ_OF
      name:NULL  type:GENERALSTRING
      name:?1  type:GENERALSTRING  value:6a6173
  name:cksum  type:SEQUENCE
    name:cksumtype  type:INTEGER  value:0x0a
    name:checksum  type:OCT_STR  value:15843a44f4f5f71746cc32e8
  name:cusec  type:INTEGER  value:0x07480d
  name:ctime  type:TIME  value:20030202162507Z
  name:authorization-data  type:SEQ_OF
    name:NULL  type:SEQUENCE
      name:ad-type  type:INTEGER
      name:ad-data  type:OCT_STR
    name:?1  type:SEQUENCE
      name:ad-type  type:INTEGER  value:0xff
      name:ad-data  type:OCT_STR  value:090000000009000000000e9ebe38d0b
      6bdca6b45b987d89f791a1
-----BEGIN SHISHI Authenticator-----
YoGDMIGAoAMCAQWhCxsJbG9jYWxob3N0ohAwDqADAgEBoQcwBRsDamFzoxcwFaAD
AgEKoQ4EDBWEOkT09fcXRswy6KQFAgMHSA2lERgPMjAwMzAyMDIxNjI1MDdaqCcw
JTAjoAMCAf+hHAQaCQAAAAAJAAAAAOnr440La9ymtFuYfYn3kaE=
-----END SHISHI Authenticator-----
```

The fourth packet contains the AP-REP, expanded as follows:

```
name:AP-REP  type:SEQUENCE
  name:pvno  type:INTEGER  value:0x05
  name:msg-type  type:INTEGER  value:0x0f
  name:enc-part  type:SEQUENCE
    name:etype  type:INTEGER  value:0x11
    name:kvno  type:INTEGER  value:0x00
    name:cipher  type:OCT_STR  value:930ccd73d8f17971460e066396228b977
    c75ad4336338f5245b09315fc21cc4e606b25abc89878d1db87fd5b208d3af9893
    0059e1c7395f49b698faac5b7fcad6ace14d2
-----BEGIN SHISHI AP-REP-----
b2IwYKADAgEFoQMCAQ+iVDBSoAMCARGhAwIBAKJGBESTDM1z2PF5cUYOBmOWIouX
fHWtQzYzj1JFsJMV/CHMTmBrJavImHjR24f9WyCNOvmJMAWeHHOV9Jtpj6rFt/yt
as4U0g==
-----END SHISHI AP-REP-----
```

After extracting the AP-REP, the EncAPRepPart is as follows:

```
name:EncAPRepPart  type:SEQUENCE
  name:ctime  type:TIME  value:20030202162507Z
  name:cusec  type:INTEGER  value:0x07480d
-----BEGIN SHISHI EncAPRepPart-----
exwwGqARGA8yMDAzMDIwMjE2MjUwN1qhBQIDB0gN
-----END SHISHI EncAPRepPart-----
```

6. Security Considerations

The authentication phase is believed to be no less secure than the Client/Server Authentication exchange described in the Kerberos 5 protocol.

If no security layer is negotiated, the connection is subject to active man-in-the-middle attackers that hijack the connection after authentication has been completed.

When security layers are used, it is believed that the communication channel negotiated by this specification is no less secure than the KRB_SAFE and KRB_PRIV primitives. In other words, it is believed that if an attack that breaches integrity or privacy of this mechanism, the same attack also applies to the Kerberos 5 specification, and vice versa.

Server implementations should be aware that the proxy function can be abused, and MAY implement precaution against this if it is considered a threat. Useful precautions include limiting the size and number of packets forwarded, and to abort the SASL exchange when the limit is reached.

Server implementations should make sure the method to look up KDC for the client indicated realm does not cause security problems. In particular, trusting unprotected DNS lookups to find the KDC of a realm may be considered as dangerous by a server.

The forward-compatibility behavior of returning empty responses to unsupported commands may be abused as a covert channel.

The reason for the client to send, in the Authenticator checksum field, not only the server random number but the entire initial server packet with the security layer bitmask and maximum cipher-text buffer size accepted by server, is to prevent an attacker from downgrading the security layer and preference for mutual authentication ultimately selected. The random number ties the client and server to the same network session, prevent man-in-the-middle attacks assuming a Kerberos 5 security layer is chosen and that the Kerberos 5 security layer is secure.

Generating AS-REP using a legacy password database requires calculating the string2key operation. This may be a costly operation (in particular for the recent AES ciphers), so servers should either pre-calculate and store the key once or take precautions against opening itself up to a Denial Of Service attack which exhausts CPU power on the server.

The security considerations of Kerberos 5 and SASL are inherited. Some immediate consequences of this follows (this is an inconclusive summary):

Note that some of the Kerberos 5 encryption types are considered weak, implementations must decide which algorithms are trusted.

Note that the encryption types indicated in AS-REQ/TGS-REQ are not integrity protected, so an attacker may downgrade the encryption keys ultimately used.

Note that Kerberos 5 do not authorize users, it only authenticate users. Applications using this mechanism must thus perform checks, not described in detail in this document, to make sure the authenticated user is authorized to the service she is requesting.

Note that the SASL framework is subject to "downgrade" attacks where an attacker forces a weaker SASL mechanism to be used. The use of, e.g., TLS [5] can be used to mitigate this.

Note that clients should use the server name exactly as the user specified, or at least abstain from canonicalizing the server name with insecure mechanisms such as unprotected DNS.

Normative References

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

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

[3] Myers, J., "Simple Authentication and Security Layer (SASL)", RFC 2222, October 1997.

Informative References

[4] Eastlake, D., Crocker, S. and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994.

[5] Dierks, T., Allen, C., Treese, W., Karlton, P., Freier, A. and

P. Kocher, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[6] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, January 2000.

Author's Address

Simon Josefsson
Drottningholmsv. 70
Stockholm 112 42
Sweden

EMail: simon@josefsson.org

Acknowledgments

Text and ideas was borrowed from the Kerberos version 4 SASL mechanism in RFC 2222. Lawrence Greenfield suggested adding a security consideration about server name canonicalization.

Intellectual Property Statement

Full Copyright Statement

Acknowledgement

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