Network Working Group A. Melnikov
Internet Draft Editor
Document: draft-ietf-sasl-rfc2222bis-08.txt June 2004
Obsoletes: RFC 2222 Expires in six months

Simple Authentication and Security Layer (SASL)

Status of this Memo

A revised version of this draft document will be submitted to the RFC editor as a Standards Track RFC for the Internet Community. Discussion and suggestions for improvement are requested. Distribution of this draft is unlimited.

When published as an RFC this document will obsolete RFC 2222.

Abstract

The Simple Authentication and Security Layer (SASL) is a framework for providing authentication and data security services in connection-oriented protocols via replaceable mechanisms. It provides structured interface between protocols and mechanisms. The resulting framework allows new protocols to reuse existing mechanisms and allows old protocols to make use of new mechanisms. The framework also provides a protocol for securing subsequent protocol exchanges within a data security layer.

This document describes how a SASL mechanism is structured, describes how protocols add support for SASL, and defines the protocol for carrying a data security layer over a connection. Additionally, this document defines one SASL mechanism, the EXTERNAL mechanism.

1. Conventions used in this document

In examples, "C:" and "S:" indicate lines sent by the client and server respectively.

The key words "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", and "MAY" in this document are to be interpreted as defined in "Key words for use in RFCs to Indicate Requirement Levels" [KEYWORDS].

Character names in this document use the notation for code points and names from the Unicode Standard [Unicode]. For example, the letter "a" may be represented as either <U+0061> or <LATIN SMALL LETTER A>.

This document uses terms "integrity protection" and "confidentiality protection". The former refers to a security layer (see Section "Introduction" below for the definition) designed to provide "data integrity service" as defined in [Sec-Glossary]. Confidentiality protection is a security layer that provides "data confidentiality service" as defined in [Sec-Glossary]. The term "confidentiality protection" usually implies "integrity protection". Security layers may offer other kinds of security services.

2. Introduction

The Simple Authentication and Security Layer (SASL) is a framework for providing authentication and data security services in connection-oriented protocols via replaceable mechanisms. SASL provides a structured interface between protocols and mechanisms. SASL also provides a protocol for securing subsequent protocol exchanges within a data security layer.

SASL design is intended to allow new protocols to reuse existing mechanisms without requiring redesign of the mechanisms and allows existing protocols to make use of new mechanisms without redesign of protocols.

The SASL is conceptually a framework which provides a layer between protocols and mechanisms, as illustrated in the following diagram.

```
             SMTP Protocol     LDAP Protocol          Other Protocols
                Profile           Profile            . . .
                       \-----        |       -----/
                             \       |      /
                              SASL framework
                             /       |      \
                       /-----        |       -----\
               DIGEST-MD5         EXTERNAL            Other Mechanisms
             SASL mechanism    SASL mechanism        . . .
```

It is through the interfaces of this layer that the framework allows any protocol to be utilized with any mechanism. While the layer does generally hide the particulars of protocols from mechanisms, the layer does not generally hide the particulars of mechanisms from protocols. For example, different mechanisms require different information to operate, some of them use password based authentication, other make use of Kerberos tickets, certificates, etc. Also, in order to perform authorization step server implementations have to implement mapping from a mechanism specific authentication identity format to a protocol specific format.

It is noted that it is possible to design and implement this framework in ways which do abstract away particulars of similar mechanisms. Such implementation could also be designed to be shared by multiple implementations of various protocols.

As illustrated above, the SASL framework interfaces with both protocols and mechanisms.

To use SASL, a protocol includes a command for identifying and authenticating a user to a server and for optionally negotiating a security layer for subsequent protocol interactions. If the use of a security layer is negotiated, that security layer is inserted between the protocol and the connection. Section 4 ("Protocol profile requirements") profiles the requirements that a protocol specification must fulfill to make use of SASL.

A SASL mechanism is a series of server challenges and client responses specific to the mechanism. Each SASL mechanism are

identity by a registered name. Section 5. ("Mechanism profile guidelines") profiles the requirements that a mechanism specification must fulfill to define a SASL mechanism.

This document is written to serve several different audiences:

o) protocol designers using this specification to support authentication in their protocol

o) mechanism designers that define new SASL mechanisms

o) implementors of clients or servers for those protocols using this specification.

The sections "Authentication Mechanisms", "Protocol Profile Requirements", "Specific Issues", and "Security Considerations" cover issues that protocol designers need to understand and address in profiling this specification for use in a specific protocol.

The sections "Authentication Mechanisms", "Mechanism Profile Requirements", "Security Considerations" and "Registration procedure" cover issues that mechanism designers need to understand and address in designing new SASL mechanisms.

The sections "Authentication Mechanisms", "Protocol profile requirements", "Specific issues" and "Security Considerations" cover issues that implementors of a protocol that uses SASL framework need to understand. The implementors will also need to understand a specification of a profile specific to the protocol, as well as aspects of mechanism specifications they intend to use (regardless of whether they are implementing the mechanisms themselves or using an existing implementation) to understand, for instance, the mechanism specific authentication identity forms, the offered services, and security and other considerations.

3. Authentication mechanisms

SASL mechanisms are named by strings, from 1 to 20 characters in length, consisting of ASCII [ASCII] upper-case letters, digits, hyphens, and/or underscores. Names of SASL mechanisms or families of mechanisms must be registered with the Internet Assigned Numbers Authority (IANA) as described in section 8.2.

The "sasl-mech" ABNF production below defines the syntax of a SASL mechanism name. This uses the Augmented Backus-Naur Form (ABNF) notation as specified in [ABNF].

```
   sasl-mech    = 1*20mech-char
   mech-char    = UPPER-ALPHA / DIGIT / HYPHEN / UNDERSCORE
                  ; mech-char is restricted to "A"-"Z", "0"-"9", "-",
                  ; and "_" from ASCII character set.

   UPPER-ALPHA  = %x41-5A
                  ; "A"-"Z"

   DIGIT        = %x30-39
                  ; "0"-"9"

   HYPHEN       = %x2D
                  ; "-"

   UNDERSCORE   = %x5F
                  ; "_"
```

3.1. Authentication protocol exchange

A SASL mechanism is responsible for conducting an authentication protocol exchange. This consists of a series of server challenges and client responses, the contents of which are specific to and defined by the mechanism. To the protocol, the challenges and responses are opaque binary tokens of arbitrary length. The protocol's profile then specifies how these binary tokens are then encoded for transfer over the connection.

After receiving an authentication command or any client response, a server mechanism may issue a challenge, indicate failure, or indicate completion. The server mechanism may return additional data with a completion indication. The protocol's profile specifies how each of these is then represented over the connection.

After receiving a challenge, a client mechanism may issue a response or abort the exchange. The protocol's profile specifies how each of these are then represented over the connection.

During the authentication protocol exchange, the mechanism performs authentication, transmits an authorization identity (frequently known as a userid) from the client to server, and negotiates the use of a mechanism-specific security layer. If the use of a security layer is agreed upon, then the mechanism must also define or negotiate the maximum security layer buffer size that each side is able to receive.

3.2. Authorization and authentication identities

SASL authentication deals with two identities: the authentication identity, derived from the client's authentication credentials; and the authorization identity, which is the result of SASL processing and is used by the server as the primary identity for making access policy decisions.

The processing model is as follows. A server, upon completion of the authentication mechanism, uses the results produced by the authentication mechanism, the client-provided authorization identity value (which may be the empty string), and local policy information to derive an authorization identity. The authorization identity is made available to further server processing for use in making access policy decisions. The provision of additional client attributes that may affect access policy is not covered by this specification.

The authorization identity may be an empty (zero length) string. In this case, the server derives an authorization identity from the client's authentication identity.

A mechanism which is incapable of transmitting an authorization identity must be treated as if it always transmits an authorization identity of an empty string.

Any normalization of the authentication identity is defined by a particular SASL mechanism, the protocol profile doesn't influence it.

The mechanism MUST preserve Unicode codepoint when transferring authorization identity (e.g. the mechanism cann't apply any form of normalization).

3.2.1. Authorization identities and proxy authentication

If the authorization identity transmitted during the authentication protocol exchange is not the empty string, this is typically referred to as "proxy authentication". This feature permits agents such as proxy servers to authenticate using their own credentials, yet request the access privileges of the identity for which they are proxying.

The server makes an implementation defined policy decision as to whether the authentication identity is permitted to have the access privileges of the authorization identity and whether the authorization identity is permitted to receive service. If it is not, the server indicates failure of the authentication protocol exchange.

As a client might not have the same information as the server, clients SHOULD NOT derive authorization identities from authentication identities. Instead, clients SHOULD provide no (or empty) authorization identity when the user has not provided an authorization identity.

The server SHOULD verify that a received authorization identity is in the correct form. Protocol profiles whose authorization identities are simple user names (e.g. IMAP [RFC 3501]) SHOULD use "SASLprep" profile [SASLprep] of the "stringprep" algorithm [StringPrep] to prepare these names for matching. The profiles MAY use a stringprep profile that is more strict than "SASLprep". If the preparation of the authorization identity fails or results in an empty string, the server MUST fail the authentication exchange. The only exception to this rule is when the received authorization identity is already the empty string.

3.2.2. Authorization Identity Format

An authorization identity is a string of zero or more Unicode [Unicode] coded characters. The NUL <U+0000> character is not permitted in authorization identities.

The character encoding scheme used for transmitting an authorization identity over the protocol is specified in each authentication mechanism. All IETF-defined mechanisms MUST, and all other mechanisms SHOULD, use UTF-8 [UTF-8]. (See [CHARSET-POLICY] for IETF policy regarding character sets and encoding schemes.)

Mechanisms are expected to be capable of carrying the entire Unicode repertoire (with the exception of the NUL character). An authorization identity of the empty string and an absent authorization identity MUST be treated as equivalent. A mechanism which provides an optional field for an authorization identity, SHOULD NOT allow that field, when present, to be empty. The meaning of the empty string as an authorization identity is described in the previous section.

3.3. Security layers

If use of a security layer is negotiated by the authentication protocol exchange, the security layer is applied to all subsequent data sent over the connection (until another security layer is negotiated; see also section 6.3). The security layer takes effect immediately following the last response of the authentication exchange for data sent by the client and the completion indication for data sent by the server. The exact position MUST be defined by the protocol profile (see section 4 part 5).

Note that all SASL mechanisms that are unable to negotiate a security layer automatically select no security layer.

Once the security layer is in effect the protocol stream is processed by the security layer into buffers of protected data. Each buffer of protected data is transferred over the connection as a stream of octets prepended with a four octet field in network byte order that represents the length of the following buffer. The length of the protected data buffer MUST be no larger than the maximum size that was either defined in the mechanism specification or negotiated by the other side during the authentication protocol exchange. Upon the receipt of a data buffer which is larger than the defined/negotiated maximal buffer size the receiver SHOULD close the connection. This might be a sign of an attack or a buggy implementation.

4. Protocol profile requirements

In order to use this specification, a protocol definition MUST supply the following information:

1) A service name, to be selected from the IANA registry of "service" elements for the GSSAPI host-based service name form [GSSAPI]. This service name is made available to the authentication mechanism.

The registry is available at the URL
<http://www.iana.org/assignments/gssapi-service-names>.

2) A definition of the command to initiate the authentication protocol exchange. This command must have as a parameter the name of the mechanism being selected by the client.

The command SHOULD have an optional parameter giving an initial response. If the protocol allows for the initial response, the protocol profile SHOULD also describe how an empty initial response is encoded. This optional parameter allows the client to avoid a round trip when using a mechanism which is defined to have the client send data first. When this initial response is sent by the client and the selected mechanism is defined to have the server start with an initial challenge, the command fails. See section 6.1 of this document for further information.

3) A definition of the method by which the authentication protocol exchange is carried out, including how the challenges and responses are encoded, how the server indicates completion or failure of the exchange, how the client aborts an exchange, and how the exchange method interacts with any line length limits in the protocol.

The exchange method SHOULD allow the server to include an optional

data ("optional challenge") with a success notification. This allows the server to avoid a round trip when using a mechanism which is defined to have the server send additional data along with the indication of successful completion. Note that an additional data with success can't be empty. See section 6.2 of this document for further information.

4) A protocol profile SHOULD specify a mechanism through which a client may obtain the names of the SASL mechanisms available to it. This is typically done through the protocol's extensions or capabilities mechanism.

5) Identification of the octet where any negotiated security layer starts to take effect, in both directions.

6) Specify if the protocol profile supports "multiple authentications" (see section 6.3).

7) If both a Transport Layer Security [TLS] and a SASL security layer are allowed to be negotiated by the protocol, the protocol profile MUST define in which order they are applied to a cleartext data sent over the connection.

8) A protocol profile MAY further refine the definition of an authorization identity by adding additional syntactic restrictions and protocol-specific semantics. A protocol profile MUST specify the form of the authorization identity (since it is protocol specific, as opposed to the authentication identity, which is mechanism specific) and how authorization identities are to be compared. Profiles whose authorization identities are simple user names (e.g. IMAP [RFC 3501]) SHOULD use "SASLprep" profile [SASLprep] of the "stringprep" algorithm [StringPrep] to prepare these names for matching. The profiles MAY use a stringprep profile that is more strict than SASLprep.

9) Where the application-layer protocol does not precisely state how identities established through SASL relate to identities used elsewhere (e.g., access controls) in the application-layer protocol, it may be useful for the application-layer protocol to provide a facility which the client may use to discover the identity used.

A protocol profile SHOULD NOT attempt to amend the definition of mechanisms or make mechanism-specific encodings. This breaks the separation between protocol and mechanism that is fundamental to the design of SASL. Likewise, SASL mechanisms SHOULD be profile neutral.

5. Mechanism profile guidelines

Designers of new SASL mechanism should be aware of the following issues:

1) Authorization identity

While some legacy mechanisms are incapable of transmitting an authorization identity (which means that for these mechanisms the authorization identity is always the empty string), newly defined mechanisms SHOULD be capable of transmitting a non-empty authorization identity. See also section 3.2.

2) Character string issues

Authentication mechanisms SHOULD encode character strings in UTF-8 [UTF-8] (see [CHARSET-POLICY] for IETF policy regarding character sets in IETF protocols). In order to avoid interoperability problems due to differing normalizations, when a mechanisms specifies that character data is to be used as input to a cryptographic and/or comparison function, the mechanism specification MUST detail how the data is to be represented, including any normalizations or other preparations, to ensure proper function. Designers of mechanisms SHOULD use the "SASLprep" profile [SASLprep] of the "stringprep" algorithm [StringPrep] where applicable. However it should be noted that this rule doesn't apply to authorization identities, as they are protocol specific.

The preparation can be potentially performed on the client end (upon getting user input or retrieving a value from configuration) or on the server end (upon receiving the value from the client, retrieving a value from its authentication database or generating a new value in order to store in in the authentication database). SASL mechanisms must define which entity (or entities) must perform the preparation. If preparation fails or results in an empty string, the entity doing the preparation MUST fail the authentication exchange.

Implementation note: A server end can be represented by multiple processes. For example, it may consist of the server process itself that communicated with a client, and a command line utility (a server agent) that is able to store passwords/hashes in a database that can be later used by the server. For the server agent the requirement to "fail the authentication exchange" should be interpreted as a requirement to refuse to store the data in the database.

3) If the underlying cryptographic technology used by a mechanism supports data integrity than the mechanism specification MUST

integrity protect the transmission of an authorization identity and the negotiation of the security layer.

4) The mechanism should not use the authorization identity in generation of any long-term cryptographic keys/hashes. The reason is that different protocols (and sometimes even different implementations of the same protocol) may use multiple forms of an authorization identity that are semantically equivalent and some clients may use one form while other clients use a different form.

5) SASL mechanisms should be designed to minimize the number of round trips required because SASL can be used with protocols where connections are short-lived.

6. Specific issues

6.1. Client sends data first

Some mechanisms specify that the first data sent in the authentication protocol exchange is from the client to the server.

If a protocol's profile permits the command which initiates an authentication protocol exchange to contain an initial client response, this parameter SHOULD be used with such mechanisms.

If the initial client response parameter is not given, or if a protocol's profile does not permit the command which initiates an authentication protocol exchange to contain an initial client response, then the server issues a challenge with no data. The client's response to this challenge is then used as the initial client response. (The server then proceeds to send the next challenge, indicates completion, or indicates failure.)

6.1.1. Client sends data first examples

The following are two examples of an SECURID authentication [SASL-SECURID] in the SMTP protocol [SMTP]. In the first example below, the client is trying fast reauthentication by sending the initial response:

```
S: 220-smtp.example.com ESMTP Server
C: EHLO client.example.com
S: 250-smtp.example.com Welcome client.example.com
S: 250-AUTH GSSAPI SECURID
S: 250 DSN
C: AUTH SECURID AG1hZ251cwAxMjM0NTY3OAA=
S: 235 Authentication successful
```

The example below is almost identical to the previous, but here the client chooses not to use the initial response parameter.

```
S: 220-smtp.example.com ESMTP Server
C: EHLO client.example.com
S: 250-smtp.example.com Welcome client.example.com
S: 250-AUTH GSSAPI SECURID
S: 250 DSN
C: AUTH SECURID
S: 334
C: AG1hZ251cwAxMjM0NTY3OAA=
S: 235 Authentication successful
```

Additonal examples that show usage of initial response can be found in section 7.2.

6.2. Server returns success with additional data

Some mechanisms may specify that additional data be sent to the client along with an indication of successful completion of the exchange. This data would, for example, authenticate the server to the client.

If a protocol's profile does not permit this additional data to be returned with a success indication, then the server issues the data as a server challenge, without an indication of successful completion. The client then responds with no data. After receiving this empty response, the server then indicates successful completion (with no additional data).

Client implementors should be aware of an additional failure case that might occur when the profile supports sending the additional data with success. Imagine that an active attacker is trying to impersonate the server and sends faked data, which should be used to authenticate the server to the client, with success. (A similar situation can happen when either the server and/or the client has a bug and they calculate different responses.) After checking the data, the client will think that the authentication exchange has failed, however the server will think that the authentication exchange has completed successfully. At this point the client can not abort the authentication exchange; it SHOULD close the connection instead. However, if the profile did not support sending of additional data with success, the client could have aborted the exchange at the very last step of the authentication exchange.

6.2.1. Server returns success with additional data examples

The following are two examples of a DIGEST-MD5 authentication [SASL-DIGEST] in the Extensible Messaging and Presence Protocol [XMPP]. In the first example below, the server is sending mutual authentication data with success.

```
C: <stream:stream
    xmlns='jabber:client'
    xmlns:stream='http://etherx.jabber.org/streams'
    to='example.com'
    version='1.0'>
S: <stream:stream
    xmlns='jabber:client'
    xmlns:stream='http://etherx.jabber.org/streams'
    id='c2s_234'
    from='example.com'
    version='1.0'>
S: <stream:features>
     <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
       <mechanism>DIGEST-MD5</mechanism>
       <mechanism>CRAM-MD5</mechanism>
     </mechanisms>
   </stream:features>
C: <auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl'
         mechanism='DIGEST-MD5'/>
S: <challenge xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
   cmVhbG09InNvbWVyZWFsbSIsbm9uY2U9Ik9BNk1HOXRFUUdtMmhoIixxb3A9
   ImF1dGgiLGNoYXJzZXQ9dXRmLTgsYWxnb3JpdGhtPW1kNS1zZXNzCg==
   </challenge>
C: <response xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
   dXNlcm5hbWU9InNvbWVub2RlIixyZWFsbT0ic29tZXJlYWxtIixub25jZT0i
   T0E2TUc5dEVRR20yaGgiLGNub25jZT0iT0E2TUhYaDZWcVRyUmsiLG5jPTAw
   MDAwMDAxLHFvcD1hdXRoLGRpZ2VzdC11cmk9InhtcHAvZXhhbXBsZS5jb20i
   LHJlc3BvbnNlPWQzODhkYWQ5MGQ0YmJkNzYwYTE1MjMyMWYyMTQzYWY3LGNo
   YXJzZXQ9dXRmLTgK
   </response>
S: <success xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
   cnNwYXV0aD1lYTQwZjYwMzM1YzQyN2I1NTI3Yjg0ZGJhYmNkZmZmZAo=
   </success>
```

The example below is almost identical to the previous, but here the server chooses not to use the additional data with success.

```
C: <stream:stream
    xmlns='jabber:client'
    xmlns:stream='http://etherx.jabber.org/streams'
```

```
          to='example.com'
          version='1.0'>
      S: <stream:stream
          xmlns='jabber:client'
          xmlns:stream='http://etherx.jabber.org/streams'
          id='c2s_234'
          from='example.com'
          version='1.0'>
      S: <stream:features>
           <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
             <mechanism>DIGEST-MD5</mechanism>
             <mechanism>CRAM-MD5</mechanism>
           </mechanisms>
         </stream:features>
      C: <auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl'
               mechanism='DIGEST-MD5'/>
      S: <challenge xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
         cmVhbG09InNvbWVyZWFsbSIsbm9uY2U9Ik9BNk1HOXRFUUdtMmhoIixxb3A9
         ImF1dGgiLGNoYXJzZXQ9dXRmLTgsYWxnb3JpdGhtPW1kNS1zZXNzCg==
         </challenge>
      C: <response xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
         dXNlcm5hbWU9InNvbWVub2RlIixyZWFsbT0ic29tZXJlYWxtIixub25jZT0i
         T0E2TUc5dEVRR20yaGgiLGNub25jZT0iT0E2TUhYaDZWcVRyUmsiLG5jPTAw
         MDAwMDAxLHFvcD1hdXRoLGRpZ2VzdC11cmk9InhtcHAvZXhhbXBsZS5jb20i
         LHJlc3BvbnNlPWQzODhkYWQ5MGQ0YmJkNzYwYTE1MjMyMWYyMTQzYWY3LGNo
         YXJzZXQ9dXRmLTgK
         </response>
      S: <challenge xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
         cnNwYXV0aD1lYTQwZjYwMzM1YzQyN2I1NTI3Yjg0ZGJhYmNkZmZmZAo=
         </challenge>
      C: <response xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>
      S: <success xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>
```

6.3. Multiple authentications

Unless otherwise stated by the protocol's profile, only one successful SASL negotiation may occur in a protocol session. In this case, once an authentication protocol exchange has successfully completed, further attempts to initiate an authentication protocol exchange fail.

If a profile explicitly permits multiple successful SASL negotiations to occur, then in no case may multiple security layers be simultaneously in effect. If a security layer is in effect and a subsequent SASL negotiation selects a second security layer, then the second security layer replaces the first. If a security layer is in effect and a subsequent SASL negotiation selects no security layer, the original security layer remains in effect.

Where a protocol profile permits multiple successful SASL negotiations, the profile MUST detail the effect of a failed SASL negotiation upon the previously established authentication state. In particular, it MUST state whether the previously established authenticated state remain in force or whether the connection is to revert to an non-authenticated state. Regardless of the specified effect upon authentication state, the previously negotiated security layer remains in effect.

7. The EXTERNAL mechanism

The mechanism name associated with external authentication is "EXTERNAL".

The client sends a single message containing the UTF-8 encoding of the authorization identity. The message may be empty. The form of the authorization identity is further restricted by the application-level protocol's SASL profile.

The server uses information, external to SASL, to determine whether the client is permitted to authenticate as the authorization identity. If the client is so authorized, the server indicates successful completion of the authentication exchange; otherwise the server indicates failure.

The system providing this external information may be, for example, IPSec [IPSec] or TLS [TLS]. However, the client can make no assumptions as to what information the server can use in determining client authorization. For example, just because TLS was established, doesn't mean that the server will use the information provided by TLS.

If the client sends the empty string as the authorization identity, the authorization identity is to be derived from authentication credentials which exist in the system that is providing the external authentication.

7.1. Formal syntax

The following syntax specification uses the augmented Backus-Naur Form (BNF) notation as specified in [ABNF]. Non-terminals referenced but not defined below are as defined by [UTF-8].

The "extern-resp" rule below defines the message sent from client to server.

```
extern-resp       = *( UTF8-char-no-nul )
```

```
   UTF8-char-no-nul  = UTF8-1-no-nul / UTF8-2 / UTF8-3 / UTF8-4

   UTF8-1-no-nul     = %x01-7F
```

7.2. Examples of SASL EXTERNAL

The following is an example of an EXTERNAL authentication in the SMTP protocol [SMTP]. In this example, the client is proxy authenticating, sending the authorization identity "fred" using in the (optional) initial response. The server has obtained the client's (authentication) identity from an external service, such as IPsec, and has a security policy that permits that identity to assume the identity of the asserted authorization identity.

To the protocol profile, the four octet sequence "fred" is an opaque binary data. The SASL protocol profile for SMTP [SMTP-AUTH] specifies that server challenges and client responses are encoded in BASE64 [BASE64, section 3]; the BASE64 encoding of "fred" is "ZnJlZA==".

```
S: 220 smtp.example.com ESMTP server ready
C: EHLO jgm.example.com
S: 250-smtp.example.com
S: 250 AUTH DIGEST-MD5 EXTERNAL
C: AUTH EXTERNAL ZnJlZA==
S: 235 Authentication successful.
```

The following example is almost identical to the one above, but the client doesn't request proxy authentication.

```
S: 220 smtp.example.com ESMTP server ready
C: EHLO jgm.example.com
S: 250-smtp.example.com
S: 250 AUTH DIGEST-MD5 EXTERNAL
C: AUTH EXTERNAL
S: 235 Authentication successful.
```

The following is an example of an EXTERNAL authentication in the IMAP4 protocol [IMAP]. IMAP4 doesn't support the initial response feature of SASL. As in the previous example, the client doesn't request proxy authentication.

```
S: * OK IMAP4rev1 Server
C: C01 CAPABILITY
S: * CAPABILITY IMAP4 IMAP4rev1 AUTH=DIGEST-MD5 AUTH=EXTERNAL
[...]
C: A01 AUTHENTICATE EXTERNAL
(note that there is a space following the "+" in the following line)
```

```
      S: +
      C:
      S: A01 OK Success
```

8. IANA Considerations

8.1. Guidelines for IANA

It is requested that IANA updates the SASL mechanisms registry as follows:

> Change the "Intended usage" of the KERBEROS_V4 and SKEY mechanism registrations to OBSOLETE. Change the "Published specification" of the EXTERNAL mechanism to this document. Updated registration information is provided in Section 8.6.

8.2. Registration procedure

Registration of a SASL mechanism is done by filling in the template in section 8.5 and sending it via electronic mail to <iana@iana.org>. IANA has the right to reject obviously bogus registrations, but will perform no review of claims made in the registration form. SASL mechanism registrations are currently available at the URL <http://www.iana.org/assignments/sasl-mechanisms>.

There is no naming convention for SASL mechanisms; any name that conforms to the syntax of a SASL mechanism name can be registered. However an IETF Standards Track document may reserve a portion of the SASL mechanism namespace ("family of SASL mechanisms") for its own use, amending the registration rules for that portion of the namespace. Each family of SASL mechanisms MUST be identified by a prefix.

While the registration procedures do not require it, authors of SASL mechanisms are encouraged to seek community review and comment whenever that is feasible. Authors may seek community review by posting a specification of their proposed mechanism as an Internet-Draft. SASL mechanisms intended for widespread use should be standardized through the normal IETF process, when appropriate.

8.3. Comments on SASL mechanism registrations

Comments on registered SASL mechanisms should first be sent to the "owner" of the mechanism and/or to the SASL WG mailing list. Submitters of comments may, after a reasonable attempt to contact the owner, request IANA to attach their comment to the SASL mechanism registration itself. If IANA approves of this, the comment will be made accessible in conjunction with the SASL mechanism registration itself.

8.4. Change control

Once a SASL mechanism registration has been published by IANA, the author may request a change to its definition. The change request follows the same procedure as the registration request.

The owner of a SASL mechanism may pass responsibility for the SASL mechanism to another person or agency by informing IANA; this can be done without discussion or review.

The IESG may reassign responsibility for a SASL mechanism. The most common case of this will be to enable changes to be made to mechanisms where the author of the registration has died, moved out of contact or is otherwise unable to make changes that are important to the community.

SASL mechanism registrations may not be deleted; mechanisms which are no longer believed appropriate for use can be declared OBSOLETE by a change to their "intended usage" field; such SASL mechanisms will be clearly marked in the lists published by IANA.

The IESG is considered to be the owner of all SASL mechanisms which are on the IETF standards track.

8.5. Registration template

Subject: Registration of SASL mechanism X

Family of SASL mechanisms: (YES or NO)

SASL mechanism name (or prefix for the family):

Security considerations:

Published specification (optional, recommended):

Person & email address to contact for further information:

Intended usage:

(One of COMMON, LIMITED USE or OBSOLETE)

Owner/Change controller:

(Any other information that the author deems interesting may be added below this line.)

8.6. The EXTERNAL mechanism registration

It is requested that the SASL Mechanism registry [IANA-SASL] entry for the EXTERNAL mechanism be updated to reflect that this document now provides its technical specification.

Subject: Updated Registration of SASL mechanism EXTERNAL

Family of SASL mechanisms: NO

SASL mechanism name: EXTERNAL

Security considerations: See RFC XXXX, section 9.

Published specification (optional, recommended): RFC XXXX

Person & email address to contact for further information:
Alexey Melnikov <Alexey.Melnikov@isode.com>

Intended usage: COMMON

Owner/Change controller: IESG <iesg@ietf.org>

Note: Updates existing entry for EXTERNAL

9. Security considerations

Security issues are discussed throughout this memo.

When the client selects a security layer with at least integrity protection, this protects against an active attacker hijacking the connection and modifying the authentication exchange to negotiate a plaintext connection.

When a server or client supports multiple authentication mechanisms, each of which has a different security strength, it is possible for an active attacker to cause a party to use the least secure mechanism

supported. To protect against this sort of attack, a client or server which supports mechanisms of different strengths should have a configurable minimum strength that it will use. It is not sufficient for this minimum strength check to only be on the server, since an active attacker can change which mechanisms the client sees as being supported, causing the client to send authentication credentials for its weakest supported mechanism.

The client's selection of a SASL mechanism is done in the clear and may be modified by an active attacker. It is important for any new SASL mechanisms to be designed such that an active attacker cannot obtain an authentication with weaker security properties by modifying the SASL mechanism name and/or the challenges and responses.

In order to detect Man-in-the-middle (MITM) attacks the client MAY list available SASL mechanisms both before and after the SASL security layer is negotiated. This allows the client to detect active attacks that remove mechanisms from the server's list of supported mechanisms, and allows the client to ensure that it is using the best mechanism supported by both client and server. New protocol profiles SHOULD require servers to make the list of SASL mechanisms available for the initial authentication available to the client after security layers are established. Some older protocols do not require this (or don't support listing of SASL mechanisms once authentication is complete); for these protocols clients MUST NOT treat an empty list of SASL mechanisms after authentication as a MITM attack.

Any protocol interactions prior to authentication are performed in the clear and may be modified by an active attacker. In the case where a client selects integrity protection, it is important that any security-sensitive protocol negotiations be performed after authentication is complete. Protocols should be designed such that negotiations performed prior to authentication should be either ignored or revalidated once authentication is complete.

When use of a security layer is negotiated by the authentication protocol exchange, the receiver should handle gracefully any protected data buffer larger than the defined/negotiated maximal size. In particular, it must not blindly allocate the amount of memory specified in the buffer size field, as this might cause the "out of memory" condition. If the receiver detects a large block, it SHOULD close the connection.

Distributed server implementations need to be careful in how they trust other parties. In particular, authentication secrets should only be disclosed to other parties that are trusted to manage and use those secrets in manner acceptable to disclosing party. Applications

using SASL assume that SASL security layers providing data confidentiality are secure even when an attacker chooses the text to be protected by the security layer. Similarly applications assume that the SASL security layer is secure even if the attacker can manipulate the ciphertext output of the security layer. New SASL mechanisms MUST meet these assumptions.

"stringprep" and Unicode security considerations apply to authentication identities, authorization identities and passwords.

The EXTERNAL mechanism provides no security protection; it is vulnerable to spoofing by either client or server, active attack, and eavesdropping. It should only be used when external security mechanisms are present and have sufficient strength.

10. References

10.1. Normative References

[ABNF] Crocker, D. (Ed.), Overell, P., "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997

[ASCII] American National Standards Institute, "Code Extension Techniques for Use with the 7-bit Coded Character Set of American National Standard Code (ASCII) for Information Interchange", FIPS PUB 35, 1974

[CHARSET-POLICY] Alvestrand, H., "IETF Policy on Character Sets and Languages", RFC 2277, BCP 18, January 1998

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

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, BCP 19, March 1997

[Unicode] The Unicode Consortium, "The Unicode Standard, Version 3.2.0" is defined by "The Unicode Standard, Version 3.0" (Reading, MA, Addison-Wesley, 2000. ISBN 0-201-61633-5), as amended by the "Unicode Standard Annex #27: Unicode 3.1" (http://www.unicode.org/reports/tr27/) and by the "Unicode Standard Annex #28: Unicode 3.2" (http://www.unicode.org/reports/tr28/).

[Stringprep] Hoffman, P., Blanchet, M., "Preparation of Internationalized Strings ("stringprep")", RFC 3454, December 2002.

[SASLprep] Zeilenga, K., "SASLprep: Stringprep profile for user names and passwords", Work in progress, draft-ietf-sasl-saslprep-XX.txt.

[UTF-8] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 3629, STD 63, November 2003.

10.2. Informative References

[SASL-GSSAPI] Melnikov, A., "SASL GSSAPI mechanisms", work in progress, draft-ietf-sasl-gssapi-XX.txt, November 2003

[SASL-DIGEST] Leach, P., Newman, C., Melnikov, A., "Using Digest Authentication as a SASL Mechanism", work in progress, draft-ietf-sasl-rfc2831bis-XX.txt, replaces RFC 2831

[SASL-OTP] Newman, C., "The One-Time-Password SASL Mechanism", RFC 2444, October 1998.

[SASL-SECURID] Nystrom, M., "The SecurID(r) SASL Mechanism", RFC 2808, April 2000.

[SMTP] Klensin, J., "Simple Mail Transfer Protocol", RFC 2821, April 2001.

[SMTP-AUTH] Myers, J., "SMTP Service Extension for Authentication", RFC 2554, March 1999.

Being revised by Siemborski, R., "SMTP Service Extension for Authentication", work in progress, draft-siemborski-rfc2554bis-XX.txt.

[XMPP] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", work in progress, draft-ietf-xmpp-core-XX.txt.

[BASE64] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 3548, July 2003.

[RFC-INSTRUCTIONS] Postel, J., Reynolds, J., "Instructions to RFC Authors", RFC 2223, October 1997.

[IANA-SASL] IANA, "SIMPLE AUTHENTICATION AND SECURITY LAYER (SASL) MECHANISMS", http://www.iana.org/assignments/sasl-mechanisms.

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

[IPSec] Kent, S., and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[Sec-Glossary] Shirey, R., "Internet Security Glossary", RFC 2828,

May 2000.

11. Editor's Address

Alexey Melnikov
Isode Limited
5 Castle Business Village
36 Station Road
Hampton, Middlesex,
TW12 2BX, United Kingdom

Email: Alexey.Melnikov@isode.com
URI: http://www.melnikov.ca/

12. Acknowledgments

This document is a revision of RFC 2222 written by John G. Myers. He also contributed significantly to this revision.

Contributions of many members of the SASL mailing list are gratefully acknowledged, in particular Kurt Zeilenga, Peter Saint-Andre, Rob Siemborski, Magnus Nystrom, Jeffrey Hutzelman, Hallvard B Furuseth, Tony Hansen, Simon Josefsson, Abhijit Menon-Sen, RL 'Bob' Morgan, Sam Hartman, Tim Alsop and Luke Howard for proofreading the document and various editorial suggestions.

13. Full Copyright Statement

Acknowledgement

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

14. Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Appendix A. Relation of SASL to transport security

Questions have been raised about the relationship between SASL and various services (such as IPsec and TLS) which provide a secured connection.

Two of the key features of SASL are:

The separation of the authorization identity from the identity in the client's credentials. This permits agents such as proxy servers to authenticate using their own credentials, yet request the access privileges of the identity for which they are proxying.

Upon successful completion of an authentication exchange, the server knows the authorization identity the client wishes to use. This allows servers to move to a "user is authenticated" state in the protocol.

These features are extremely important to some application protocols, yet Transport Security services do not always provide them. To define SASL mechanisms based on these services would be a very messy task, as the framing of these services would be redundant with the framing of SASL and some method of providing these important SASL features would have to be devised.

Sometimes it is desired to enable within an existing connection the use of a security service which does not fit the SASL model. (TLS is an example of such a service.) This can be done by adding a command, for example "STARTTLS", to the protocol. Such a command is outside the scope of SASL, and should be different from the command which starts a SASL authentication protocol exchange.

In certain situations, it is reasonable to use SASL underneath one of these Transport Security services. The transport service would secure the connection, either service would authenticate the client, and SASL would negotiate the authorization identity. The SASL negotiation would be what moves the protocol from "unauthenticated" to "authenticated" state. The "EXTERNAL" SASL mechanism is explicitly intended to handle the case where the transport service secures the connection and authenticates the client and SASL negotiates the authorization identity.

Appendix B. Relationship to other documents

This document obsoletes RFC 2222. It replaces all portions of RFC 2222 excepting sections 7.1 (Kerberos version 4 mechanism), 7.2 (GSSAPI mechanism), 7.3 (S/Key mechanism). The Kerberos version 4 (KERBEROS_IV) and S/Key (SKEY) mechanisms are now viewed as obsolete. The GSSAPI mechanism is now separately specified [SASL-GSSAPI].

Appendix C. Changes since RFC 2222

The GSSAPI mechanism was removed. It is now specified in a separate document [SASL-GSSAPI].

The "KERBEROS_V4" mechanism defined in RFC 2222 is obsolete and has

been removed.

The "SKEY" mechanism described in RFC 2222 is obsolete and has been removed. It has been replaced by the OTP mechanism [SASL-OTP].

The introduction has been substantially reorganized and clarified.

Clarified the definition and semantics of the authorization identity.

Prohibited the NUL character in authorization identities.

Added a section on character string issues.

The word "must" in the first paragraph of the "Protocol profile requirements" section was changed to "MUST".

Specified that protocol profiles SHOULD provide a way for clients to discover available SASL mechanisms.

Made the requirement that protocol profiles specify the semantics of the authorization identity optional to the protocol profile. Clarified that such a specification is a refinement of the definition in the base SASL spec.

Added a requirement discouraging protocol profiles from breaking the separation between protocol and mechanism.

Mentioned that standards track documents may carve out their own portions of the SASL mechanism namespace and may amend registration rules for the portion. However registration of individual SASL mechanisms is still required.

Specified that the authorization identity in the EXTERNAL mechanism is encoded in UTF-8.

Added a statement that a protocol profile SHOULD allow challenge data to be sent with a success indication.

Added a security consideration for the EXTERNAL mechanism.

Clarified sections concerning success with additional data.

Cleaned up IANA considerations/registrations and assembled them in one place.

Updated references and split them into Informative and Normative.

Added text to the Security Considerations section regarding handling

of extremely large SASL blocks.

Replaced UTF-8 ABNF with the reference to the UTF-8 document.

Added text about SASLprep for authentication identities and passwords. Described where SASLprep preparation should take place.

Added paragraph about verifying authorization identities.

Added a protocol profile requirement to specify interaction between SASL and TLS security layers.

Added a protocol profile requirement to specify if it supports reauthentication.

Removed the text that seemed to suggest that SASL security layer must not be used when TLS is available.

Created two subsections in 3.2 to talk separately about proxy authorization and format of the authorization identities.

Made requirement to verify that an authorization identity is correct by performing SASLprep a SHOULD, instead of a MUST.

Clarified that each SASL mechanism must decide where SASLprep is taking place.

Added 4 new examples for initial response and additional data with success.

Added text on checking the list of available SASL mechanisms after negotiating a security layer.

Added definition of "integrity protection" and "confidentiality protection".

Added warning about negotiating no layer once a security layer is negotiated.

Added new section with guidelines to a SASL mechanism designer.

Added a requirement to specify how an empty initial challenge is encoded if initial response is supported by a protocol.

Clarified that empty "additional data with success" is not allowed.

Replaced "buffers of security encoded data" with "buffers of protected data" for clarity.

Clarified that SASL EXTERNAL can be used even with SASL profiles that don't support initial data with success.