RFC 222 - Simple Authentication and Security Layer (SASL)

Network Working Group J. Myers
Request for Comments: 2222 Netscape Communications
Category: Standards Track October 1997

Simple Authentication and Security Layer (SASL)

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (1997). All Rights Reserved.

Table of Contents

1. Abstract 2
2. Organization of this Document 2
2.1. How to Read This Document 2
2.2. Conventions Used in this Document 2
2.3. Examples 3
3. Introduction and Overview 3
4. Profiling requirements 4
5. Specific issues 5
5.1. Client sends data first 5
5.2. Server returns success with additional data 5
5.3. Multiple authentications 5
6. Registration procedures 6
6.1. Comments on SASL mechanism registrations 6
6.2. Location of Registered SASL Mechanism List 6
6.3. Change Control 7
6.4. Registration Template 7
7. Mechanism definitions 8
7.1. Kerberos version 4 mechanism 8
7.2. GSSAPI mechanism 9
7.2.1 Client side of authentication protocol exchange 9
7.2.2 Server side of authentication protocol exchange 10
7.2.3 Security layer 11
7.3. S/Key mechanism 11
7.4. External mechanism 12
8. References 13
9. Security Considerations 13
10. Author's Address 14

Appendix A. Relation of SASL to Transport Security 15
Full Copyright Statement 16

1. Abstract

This document describes a method for adding authentication support to

connection-based protocols. To use this specification, a protocol includes a command for identifying and authenticating a user to a server and for optionally negotiating protection of subsequent protocol interactions. If its use is negotiated, a security layer is inserted between the protocol and the connection. This document describes how a protocol specifies such a command, defines several mechanisms for use by the command, and defines the protocol used for carrying a negotiated security layer over the connection.

2. Organization of this Document

2.1. How to Read This Document

This document is written to serve two different audiences, protocol designers using this specification to support authentication in their protocol, and implementors of clients or servers for those protocols using this specification.

The sections "Introduction and Overview", "Profiling requirements", and "Security Considerations" cover issues that protocol designers need to understand and address in profiling this specification for use in a specific protocol.

Implementors of a protocol using this specification need the protocol-specific profiling information in addition to the information in this document.

2.2. 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" [RFC 2119].

2.3. Examples

Examples in this document are for the IMAP profile [RFC 2060] of this specification. The base64 encoding of challenges and responses, as well as the "+ " preceding the responses are part of the IMAP4 profile, not part of the SASL specification itself.

3. Introduction and Overview

The Simple Authentication and Security Layer (SASL) is a method for adding authentication support to connection-based protocols. To use this specification, 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.

The command has a required argument identifying a SASL mechanism. SASL mechanisms are named by strings, from 1 to 20 characters in length, consisting of upper-case letters, digits, hyphens, and/or underscores. SASL mechanism names must be registered with the IANA. Procedures for registering new SASL mechanisms are given in the section "Registration procedures"

If a server supports the requested mechanism, it initiates an authentication protocol exchange. This consists of a series of server challenges and client responses that are specific to the requested mechanism. The challenges and responses are defined by the mechanisms as 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 the authentication command or any client response, a server may issue a challenge, indicate failure, or indicate completion. The protocol's profile specifies how the server indicates which of the above it is doing.

After receiving a challenge, a client may issue a response or abort the exchange. The protocol's profile specifies how the client indicates which of the above it is doing.

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 cipher-text buffer size that each side is able to receive.

The transmitted authorization identity may be different than the identity in the client's authentication 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. With any mechanism, transmitting an authorization identity of the empty string directs the server to derive an authorization identity from the client's authentication credentials.

If use of a security layer is negotiated, it is applied to all subsequent data sent over the connection. 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. Once the security layer is in effect, the protocol stream is processed by the security layer into buffers of cipher-text. Each buffer 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 cipher-text buffer must be no larger than the maximum size that was defined or negotiated by the other side.

4. Profiling 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 [RFC 2078].

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

   The command SHOULD have an optional parameter giving an initial response. 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 5.1 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.

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

5. A specification of how the authorization identity passed from the client to the server is to be interpreted.

5. Specific issues

5.1. 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.)

5.2. Server returns success with additional data

Some mechanisms may specify that server challenge 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 server challenge to be

returned with a success indication, then the server issues the 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.

5.3. 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.

In the case that 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 no security layer, the original security layer remains 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.

6. Registration procedures

Registration of a SASL mechanism is done by filling in the template in section 6.4 and sending it in to iana@isi.edu. IANA has the right to reject obviously bogus registrations, but will perform no review of clams made in the registration form.

There is no naming convention for SASL mechanisms; any name that conforms to the syntax of a SASL mechanism name can be registered.

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.

6.1. Comments on SASL mechanism registrations

Comments on registered SASL mechanisms should first be sent to the "owner" of the mechanism. 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.

6.2. Location of Registered SASL Mechanism List

SASL mechanism registrations will be posted in the anonymous FTP directory "ftp://ftp.isi.edu/in-notes/iana/assignments/sasl-mechanisms/" and all registered SASL mechanisms will be listed in the periodically issued "Assigned Numbers" RFC [currently STD 2, RFC 1700]. The SASL mechanism description and other supporting material may also be published as an Informational RFC by sending it to "rfc-

editor@isi.edu" (please follow the instructions to RFC authors [RFC 2223]).

6.3. 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 use" 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.

6.4. Registration Template

To: iana@iana.org
Subject: Registration of SASL mechanism X

SASL mechanism name:

Security considerations:

Published specification (optional, recommended):

Person & email address to contact for further information:

Intended usage:

(One of COMMON, LIMITED USE or OBSOLETE)

Author/Change controller:

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

7. Mechanism definitions

The following mechanisms are hereby defined.

7.1. Kerberos version 4 mechanism

The mechanism name associated with Kerberos version 4 is "KERBEROS_V4".

The first challenge consists of a random 32-bit number in network byte order. The client responds with a Kerberos ticket and an authenticator for the principal "service.hostname@realm", where "service" is the service name specified in the protocol's profile, "hostname" is the first component of the host name of the server with all letters in lower case, and where "realm" is the Kerberos realm of the server. The encrypted checksum field included within the Kerberos authenticator contains the server provided challenge in network byte order.

Upon decrypting and verifying the ticket and authenticator, the server verifies that the contained checksum field equals the original server provided random 32-bit number. Should the verification be successful, the server must add one to the checksum and construct 8 octets of data, with the first four octets containing the incremented checksum in network byte order, the fifth octet containing a bit-mask specifying the security layers supported by the server, and the sixth through eighth octets containing, in network byte order, the maximum cipher-text buffer size the server is able to receive. The server must encrypt using DES ECB mode the 8 octets of data in the session key and issue that encrypted data in a second challenge. The client considers the server authenticated if the first four octets of the un-encrypted data is equal to one plus the checksum it previously sent.

The client must construct data with the first four octets containing the original server-issued checksum in network byte order, the fifth octet containing the bit-mask specifying the selected security layer, the sixth through eighth octets containing in network byte order the maximum cipher-text buffer size the client is able to receive, and the following octets containing the authorization identity. The client must then append from one to eight zero-valued octets so that the length of the data is a multiple of eight octets. The client must then encrypt using DES PCBC mode the data with the session key and respond with the encrypted data. The server decrypts the data and verifies the contained checksum. The server must verify that the principal identified in the Kerberos ticket is authorized to connect as that authorization identity. After this verification, the authentication process is complete.

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

1 No security layer
2 Integrity (krb_mk_safe) protection

4 Privacy (krb_mk_priv) protection

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

EXAMPLE: The following are two Kerberos version 4 login scenarios to the IMAP4 protocol (note that the line breaks in the sample authenticators are for editorial clarity and are not in real authenticators)

```
S: * OK IMAP4 Server
C: A001 AUTHENTICATE KERBEROS_V4
S: + AmFYig==
C: BAcAQU5EUkVXLkNNVS5FRFUAOCAsho84kLN3/IJmrMG+25a4DT
   +nZImJjnTNHJUtxAA+o0KPKfHEcAFs9a3CL5Oebe/ydHJUwYFd
   WwuQ1MWiy6IesKvjL5rL9WjXUb9MwT9bpObYLGOKi1Qh
S: + or//EoAADZI=
C: DiAF5A4gA+oOIALuBkAAmw==
S: A001 OK Kerberos V4 authentication successful


S: * OK IMAP4 Server
C: A001 AUTHENTICATE KERBEROS_V4
S: + gcfgCA==
C: BAcAQU5EUkVXLkNNVS5FRFUAOCAsho84kLN3/IJmrMG+25a4DT
   +nZImJjnTNHJUtxAA+o0KPKfHEcAFs9a3CL5Oebe/ydHJUwYFd
   WwuQ1MWiy6IesKvjL5rL9WjXUb9MwT9bpObYLGOKi1Qh
S: A001 NO Kerberos V4 authentication failed
```

7.2. GSSAPI mechanism

The mechanism name associated with all mechanisms employing the GSSAPI [RFC 2078] is "GSSAPI".

7.2.1 Client side of authentication protocol exchange

The client calls GSS_Init_sec_context, passing in 0 for input_context_handle (initially) and a targ_name equal to output_name from GSS_Import_Name called with input_name_type of GSS_C_NT_HOSTBASED_SERVICE and input_name_string of "service@hostname" where "service" is the service name specified in the protocol's profile, and "hostname" is the fully qualified host name of the server. The client then responds with the resulting output_token. If GSS_Init_sec_context returns GSS_S_CONTINUE_NEEDED,

then the client should expect the server to issue a token in a subsequent challenge. The client must pass the token to another call to GSS_Init_sec_context, repeating the actions in this paragraph.

When GSS_Init_sec_context returns GSS_S_COMPLETE, the client takes the following actions: If the last call to GSS_Init_sec_context returned an output_token, then the client responds with the output_token, otherwise the client responds with no data. The client should then expect the server to issue a token in a subsequent challenge. The client passes this token to GSS_Unwrap and interprets the first octet of resulting cleartext as a bit-mask specifying the security layers supported by the server and the second through fourth octets as the maximum size output_message to send to the server. The

client then constructs data, with the first octet containing the bit-mask specifying the selected security layer, the second through fourth octets containing in network byte order the maximum size output_message the client is able to receive, and the remaining octets containing the authorization identity. The client passes the data to GSS_Wrap with conf_flag set to FALSE, and responds with the generated output_message. The client can then consider the server authenticated.

7.2.2 Server side of authentication protocol exchange

The server passes the initial client response to GSS_Accept_sec_context as input_token, setting input_context_handle to 0 (initially). If GSS_Accept_sec_context returns GSS_S_CONTINUE_NEEDED, the server returns the generated output_token to the client in challenge and passes the resulting response to another call to GSS_Accept_sec_context, repeating the actions in this paragraph.

When GSS_Accept_sec_context returns GSS_S_COMPLETE, the client takes the following actions: If the last call to GSS_Accept_sec_context returned an output_token, the server returns it to the client in a challenge and expects a reply from the client with no data. Whether or not an output_token was returned (and after receipt of any response from the client to such an output_token), the server then constructs 4 octets of data, with the first octet containing a bit-mask specifying the security layers supported by the server and the second through fourth octets containing in network byte order the maximum size output_token the server is able to receive. The server must then pass the plaintext to GSS_Wrap with conf_flag set to FALSE and issue the generated output_message to the client in a challenge. The server must then pass the resulting response to GSS_Unwrap and interpret the first octet of resulting cleartext as the bit-mask for the selected security layer, the second through fourth octets as the maximum size output_message to send to the client, and the remaining

octets as the authorization identity. The server must verify that the src_name is authorized to authenticate as the authorization identity. After these verifications, the authentication process is complete.

7.2.3 Security layer

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

1 No security layer
2 Integrity protection.
  Sender calls GSS_Wrap with conf_flag set to FALSE
4 Privacy protection.
  Sender calls GSS_Wrap with conf_flag set to TRUE

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

7.3. S/Key mechanism

The mechanism name associated with S/Key [RFC 1760] using the MD4 digest algorithm is "SKEY".

The client sends an initial response with the authorization identity.

The server then issues a challenge which contains the decimal sequence number followed by a single space and the seed string for the indicated authorization identity. The client responds with the one-time-password, as either a 64-bit value in network byte order or encoded in the "six English words" format.

The server must verify the one-time-password. After this verification, the authentication process is complete.

S/Key authentication does not provide for any security layers.

EXAMPLE: The following are two S/Key login scenarios in the IMAP4 protocol.

```
S: * OK IMAP4 Server
C: A001 AUTHENTICATE SKEY
S: +
C: bW9yZ2Fu
S: + OTUgUWE1ODMwOA==
C: Rk9VUiBNQU5OIFNPT04gRklSIFZBUlkgTUFTSA==
S: A001 OK S/Key authentication successful
```



```
S: * OK IMAP4 Server
C: A001 AUTHENTICATE SKEY
S: +
C: c21pdGg=
S: + OTUgUWE1ODMwOA==
C: BsAY3g4gBNo=
S: A001 NO S/Key authentication failed
```

The following is an S/Key login scenario in an IMAP4-like protocol which has an optional "initial response" argument to the AUTHENTICATE command.

```
S: * OK IMAP4-Like Server
C: A001 AUTHENTICATE SKEY bW9yZ2Fu
S: + OTUgUWE1ODMwOA==
C: Rk9VUiBNQU5OIFNPT04gRklSIFZBUlkgTUFTSA==
S: A001 OK S/Key authentication successful
```

7.4. External mechanism

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

The client sends an initial response with the authorization identity.

The server uses information, external to SASL, to determine whether the client is authorized 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 or TLS.

If the client sends the empty string as the authorization identity (thus requesting the authorization identity be derived from the client's authentication credentials), the authorization identity is to be derived from authentication credentials which exist in the system which is providing the external authentication.

8. References

[RFC 2060] Crispin, M., "Internet Message Access Protocol - Version 4rev1", RFC 2060, December 1996.

[RFC 2078] Linn, J., "Generic Security Service Application Program Interface, Version 2", RFC 2078, January 1997.

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

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

[RFC 1760] Haller, N., "The S/Key One-Time Password System", RFC 1760, February 1995.

[RFC 1700] Reynolds, J., and J. Postel, "Assigned Numbers", STD 2, RFC 1700, October 1994.

9. Security Considerations

Security issues are discussed throughout this memo.

The mechanisms that support integrity protection are designed such that the negotiation of the security layer and authorization identity is integrity protected. 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.

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.

10. Author's Address

John G. Myers
Netscape Communications
501 E. Middlefield Road
Mail Stop MV-029
Mountain View, CA 94043-4042

EMail: jgmyers@netscape.com

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:

1. 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.

2. 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.

When using SASL underneath a sufficiently strong Transport Security service, a SASL security layer would most likely be redundant. The client and server would thus probably want to negotiate off the use of a SASL security layer.

Full Copyright Statement

Copyright (C) The Internet Society (1997). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implmentation may be prepared, copied, published andand distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.