Abstract

SASL is used for authentication in many application protocols. This specification extends it to allow credentials from a home realm to be used against external services. To this end, it introduces a secure end-to-end wrapper for SASL traffic and a link back from to the home realm based on Diameter.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 26, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Introduction

It is common for Internet users to combine services from a variety of providers. Along with this, an ad hoc practice has arisen of using the local identity schemes of these providers. These are not integrated, and the practice tends to reduce the control of users over their online identity. A solution to this is generic support for realm crossover, where an externally acquired service can make a callback to a home realm to authenticate a user’s identity and use that for service-specific authorisation.

SASL [RFC4422] is instrumental in authentication across a wide range of application protocols; it allows those protocols to abstract from the actual authentication mechanisms, and at the same time it allows authentication mechanisms to not be concerned about the application protocol. SASL can easily be funneled from one protocol into another, modulo a number of security concerns.

Diameter and its Network Access application are instrumental in authenticating a user under a realm, while not providing the resources that an application protocol would imply. Moreover, Diameter service can be declared under a domain name in a manner that is standardised, scalable and secure.
This allows a foreign server to authenticate a client to authenticate with its home realm:

```
+--------+    SASL     +--------+    SASL     +--------+
| Client |-----------> | Server | ---------> |  Realm  |
+--------+  AppProto   +--------+  Diameter  +---------+
||                     ||                    ||
john@example.com        find SRV, TLSA          example.com
& credential            relay SASL           authentication
```

Diameter can send a mere notification of authentication, and the foreign server can use DANE [RFC6698] to validate the origin of these notification. Diameter in the foreign server will authenticate to the home realm, which may then decide to add resources beyond the basic notification of authentication.

SASL mechanisms are not generally protected against attacks by men in the middle named Eve. This is usually compensated for by wrapping the application protocol in TLS, but since that would only protect one leg of the intended realm crossover, this raises a need for end-to-end encryption. This can be established along with other credentials for the home realm, but an end-to-end mechanism needs to be defined. This specification introduces a wrapper for that purpose, and nests a SASL exchange with the home realm under its cloak.

Finally, to avoid the use of one authentication exchange to validate another, it is advisable to incorporate channel binding [RFC5056] [RFC5801] when making use of backends. When passing SASL tokens between application protocol and Diameter backend, the channel binding information from the application protocol would be supplied as a side-note to the Diameter backcall.

2. The SASL crossover mechanism SXOVER

SXOVER is a SASL authentication mechanism that encrypts all information between a SASL client and SASL server, except for the realm name to which they direct the authentication. The realm can be used by an foreign server to redirect SXOVER to a home realm, for instance using Diameter. SXOVER does not reveal success or failure to this foreign server, but Diameter would release this information in a manner that requires no knowledge of the SASL exchange.

The first SXOVER message supplies a session key to the SASL server. The server responds with a list of SASL mechanisms to be used under the cloak of the session key. Then, the client selects a mechanism and the customary exchange follows, but under protection of the session key.
Certain information in SXOVER is encrypted with the encrypt operation [Section 3 of [RFC3961]] with default initial state; this always includes integrity protection. The value for key usage is KIP_KEYUSAGE_MAPPING or 1864 for the long-term key and KIP_KEYUSAGE_USERDATA of 1863 for the session key; these values and the formats exchanged are compatible with the Keyful Identity Protocol, which may or may not develop independently of SXOVER.

2.1. SXOVER initial response

The SXOVER exchange starts with an initial response message, traveling along the selection of SXOVER as the SASL mechanism. This initial response contains the following:

SXOVER-Initial-Response ::= SEQUENCE {
   realm IA5String, -- Lowercase domain name, no trailing dot
   inictr Counter, -- Initial counter value
   keyno KeyNumber, -- With realm and encalg, identifies...
   encalg EncryptAlg, -- ...the key for svckeymud decryption
   seskey OCTET STRING -- RFC 3961 encrypted, key usage 1864
}

Counter ::= INTEGER (0..4294967295) -- Unsigned 32-bit
KeyNumber ::= INTEGER (0..4294967295) -- Unsigned 32-bit
EncryptAlg ::= INTEGER (-2147483648..2147483647) -- Signed 32-bit

The one value of interest to the foreign server is the realm, which it needs to determine the home realm of the client. It finds the Diameter service underneath, and starts passing this SASL message and any that follow between the endpoints. Note that this behavior is specific to the SXOVER mechanism; it is just as well possible for a foreign server to welcome ANONYMOUS clients, which it can handle locally.

The inictr value is used as a bit of entropy, thus making it more difficult to mount replay attacks, though not at the level of security that a proper SASL mechanism can achieve through dynamic challenges and/or channel binding.

The keyno and encalg values present identification information for a key at the Diameter/SASL server, and the seskey is a representation of a session key suitable for decryption with that identified key.

The value of the seskey MAY be locally determined, but by default it SHOULD be a random seed that can serve as input to the random-to-key function with the required key-generation seed-length [Section 3 of [RFC3961]] for a session key with the same encryption algorithm enctype as the identified key. The random seed is protected by
encryption to the identified key using the Encrypt function [Section 3 of [RFC3961]], which always involves authenticity. The key usage number is shared from the independent KIP protocol, and is set to KIP_KEYUSAGE_MAPPING or 1864.

2.2. SXOVER initial challenge

The initial SXOVER challenge is a server’s response in which it presents the choice of mechanism names to use under the cloak of SXOVER. It does not present any other information. The following information is sent as one block under protection of the seskey:

SXOVER-Initial-Challenge ::= SEQUENCE {
  ctr           Counter,                 -- Counter value is inictr+1
  realm         IA5String,               -- Confirms the realm securely
  mechlist      SEQUENCE OF IA5String,   -- Available SASL mechanisms
  chanbindmth   IA5String,               -- TODO:REALLY?
                 -- Method of channel binding
  chanbindval   OCTET STRING             -- TODO:REALLY?
                 -- Value for channel binding
}

This message precedes the SXOVER wrapping of a SASL exchange by first passing the inner SASL mechanisms, which cannot be taken from the list provided by the foreign server. The mechanisms listed here are specific for the home realm of the client, information that could not be known to the foreign server before learning about the targeted realm.

The ctr value is simply inictr incremented by 1, with a wrap-around to stay within an unsigned 32-bit range. It MUST be validated by the SASL client.

The realm is repeated from the SXOVER request, but this time it is protected by the session key. Therefore, the SASL client MUST validate it before starting the wrapped SASL exchange.

The mechlist informs the SASL client of the mechanisms available for authentication against the SASL server. These can be used for the wrapped SASL exchange. The list is not related to any mechanism list that the foreign server will have sent before. Specifically, SXOVER and ANONYMOUS mechanisms should not occur in the wrapped mechlist.

TODO: The chanbindmth and chanbindval relay what channel binding information the SASL server obtained, and which can be used by the SASL client if it wants to continue. These values MUST be verified against the actual context first, to ensure that they do represent security requirements. If a mechanism is used that does not include
channel binding, then these fields can add some channel binding, but
it is invariably better to employ channel binding that is
cryptographically bound to the authentication operation.

2.3. SXOVER responses

Further SXOVER responses are essentially SASL responses and initial
responses, encrypted under the seskey, but there is one exception;
the first response must select a SASL mechanism. There is a separate
 provision for sending no data, distinguishable from empty data, if
this is desired by the SASL mechanism:

SXOVER-Response ::= SEQUENCE {
  ctr     Counter,       -- 1 + Previous ctr value
  c2s     SaslToken,     -- NULL or token, client to server
  mechsel IA5String OPTIONAL  -- SASL mechanism name selection
}

SaslToken ::= CHOICE {
  token   OCTET STRING,
  no-token NULL
}

This is the first and later of the wrapped SASL messages sent from
the client to the server. When it is the first, the mechsel field
MUST specify the SASL mechanism that the client selected from the
mechsel issued before. In any message, the ctr value is one more
than the value in the previous SASL message, reduced to an unsigned
32-bit range.

The SaslToken in c2s is either a literal OCTET STRING with the SASL
token to pass, or it is NULL if no token is passed. This implements
a distinction between an empty token and no token, as required for
SASL.

2.4. SXOVER challenges

Further SXOVER challenges are essentially SASL challenges and initial
challenges, encrypted under the seskey. There is a separate
 provision for sending no data, distinguishable from empty data, if
this is desired by the SASL mechanism:

SXOVER-Challenge ::= SEQUENCE {
  ctr  Counter,       -- 1 + Previous ctr value
  s2c  SaslToken,     -- NULL or token, server to client
  extra OCTET STRING OPTIONAL  -- On success, optional extra token
}
This is the first and later of the wrapped SASL messages sent from the server to the client. In any message, the ctr value is one more than the value in the previous SASL message, reduced to an unsigned 32-bit range.

The SaslToken in s2c is either a literal OCTET STRING with the SASL token to pass, or it is NULL if no token is passed. This implements a distinction between an empty token and no token, as required for SASL.

The extra value can be passed along as a hint to the user for a successful authentication. Mechanisms do not commonly use the field, but SASL offers it. The distinction between an empty OCTET STRING and an absent value is assured through the OPTIONAL modifier. Note that this value should not be passed as part of the SXOVER exchange, as it is part of the SASL mechanism that was selected with mechsel in the wrapped exchange. SXOVER does not specify an extra value, so the field in the outer SASL exchange that runs SXOVER will not be used.

3. Embedding SASL in Diameter

SASL messages in Diameter use a number of AVPs [RFC6733] that are defined for this purposes. They occur in those combinations that are defined for SASL.

SASL over Diameter can only be used to relay the SXOVER mechanism to a home realm. This means that no negotiation of mechanisms is needed at the Diameter level; this is handled under the SXOVER cloak. The same holds for any negotiation of channel binding; it is part of the cloacked SASL exchange.

3.1. AVP Definitions for SASL

These AVPs are added to the set that is used with the Network Access application, and can therefore be used in AA-Request and AA-Answer messages. On top of that, the SASL-Mechanisms AVP may also occur in a Capabilities Exchange Answer. The User-Name AVP MUST be supplied in the AA-Answer to inform the server about the user name that the backend decided on; the server MAY send a hint requesting a value in the User-Name AVP in the AA-Request.

3.1.1. SASL-Mechanism

The SASL-Mechanism AVP has AVP Code TBD0. This specification only uses the mechanism name SXOVER as a value for this AVP. It MUST be included in the first message of an SXOVER exchange sent to the home realm, and it SHOULD be verified by the home realm upon reception.
Its purpose is mostly to distinguish this specification from potential future specifications to encapsulate SASL in Diameter.

Though not used in this specification, this AVP may also be supplied from the home realm to the Diameter client to hold a space-separated list of SASL mechanisms.

3.1.2. SASL-Token

The SASL-Token AVP has AVP Code TBD1. Note that SASL requires distinction between empty and absent tokens; absent SASL tokens are represented by absence of the SASL-Token AVP and empty SASL tokens are represented as a present SASL-Token AVP with zero content bytes.

3.1.3. SASL-Channel-Binding

The SASL-Channel-Binding AVP has AVP Code TBD2. It SHOULD appear along the first SASL-Token AVP for a Network Access session. The AVP may occur more than once, to indicate support of multiple forms of channel binding.

When the client connects to the foreign service over TLS, the tls-unique form [RFC5929] of channel binding is RECOMMENDED. Specific foreign servers may however be exempted by the home realm.

The contents of this AVP are:

name is the standard name of the channel binding information, followed by a zero-valued byte.

value contains the bytes of the channel binding information.

Normally, channel binding information should be sourced from the underlying communications channel, but this information is not available to backend running Diameter. To enable channel binding between the end points, the foreign server incorporates the channel binding information that the client can use in its connection to the foreign server. This is useful to mitigate replay attacks, which is why its use is RECOMMENDED. Channel binding provides better guarantees than the simple inictr/ctr mechanism used in SXOVER.

4. Running Diameter as a SASL Backend

Following are a few practical considerations in relation to the Diameter connectivity for SASL.
4.1. Diameter is an SCTP service

Diameter is primarily an SCTP-based protocol [RFC6733], for reasons of scalability and efficiency. SASL Diameter benefits from these properties and embraces the SCTP carrier. Operating system support for SCTP is wide-spread, but parts of network infrastructure may not support it, and that may cause implementations to add a fallback to more traditional protocols. Standards offer two options for doing this.

Diameter can fallback to run over TCP, which is mostly of use to client-only machines, but it sacrifices several benefits of the SCTP carrier. Since the SASL Diameter embedding typically involves no client systems, this option is NOT RECOMMENDED.

SCTP may be run over a UDP transport using port 9899 [RFC6951], which does not sacrifice much; it only inserts a UDP header before each message. This is a reasonable expectation of foreign servers as well as home realms, so this additional option is RECOMMENDED for situations where a fallback for plain SCTP is desired. It is standardised as a socket option SCTP_REMOTE_UDP_ENCAPS_PORT, and only involves a small repetition in code, with a minor change between the attempts.

4.2. Reliance on DANE and DNSSEC

Diameter always involves the use of TLS, but there is a number of choices concerning the validation of connections through DNSSEC and DANE. It is the home realm’s prerogative what level of protection it upholds for its client identities, but any foreign server can choose to raise the bar by setting a minimum standard.

DNSSEC is a useful protection mechanism for the _diameter._sctp SRV records that lead to the Diameter host and its port for the home realm. This does not protect against forged IP addresses, port mappings or routing. To protect against this as well, a TLSA record for the service host and port, along with the _sctp protocol label, should be used as specified for DANE [RFC6698].

Home realms that choose to be light on such measures risk that identities are forged, in spite of their use of TLS. Foreign servers MAY choose to reject such home realms, or alternatively be more inquisitive about the certificates used.
4.3. Foreign Service SASL Mechanisms

A foreign server MUST offer SXOVER if it wants to support realm crossover via Diameter as specified herein. In addition, it MAY offer SASL mechanisms that it resolves locally.

The ANONYMOUS method for SASL [RFC4505] may be offered for guest access. The PLAIN method [RFC4616] continues to be ill-advised, especially with modern methods such as SCRAM [RFC5802] to address the needs of local accounts with password validation.

The HTTP protocol does not yet support SASL, and it is not optimal from a security viewpoint to integrate credentials in the dynamic environment of HTML, where dynamic content from potentially undesirable origins come together in a manner not controllable to the end user. One remedy is to use HTTP and its authentication methods that match with SASL, such as SCRAM for HTTP [RFC7804]. Another remedy is to switch to generic SASL embedding in HTTP [TODO:REF: draft-vanrein-httpauth-sasl] and gain replay protection through channel binding.

Many application protocols offer richer semantics than HTTP, making them better targets for automation. Their reliance on SASL has made them less tractable as a service to third parties. One reason for introducing SXOVER is in the hope to make it possible to have those semantically rich applications as a third-party offering.

5. Security Considerations

From the perspective of the client and the home realm, the safety of the SASL credentials is paramount. Since not all SASL mechanisms are safe from inspection by the foreign server, and since TLS cannot help there either, there is a need for some caution.

The limitation of the Diameter carrier for SASL to SXOVER reduces this risk, by only authenticating SASL mechanisms under end-to-end encryption between the client and home realm. It is generally understood that clients must not send unprotected SASL authentication attempts to arbitrary parties, but SXOVER adds a facility that is safe for clients to use in this manner. The SXOVER mechanism could even be used without TLS protection.

From the perspective of the foreign server, the security concern is to be certain of an identity. The home realm sends this information back when SXOVER authentication succeeds, and the communication doing so is protected with TLS. The certificate of the Diameter server can be validated, and for cautious home realms there could be an additional check based on DANE.
6. IANA Considerations

This specification defines three AVP Codes for use with Diameter. IANA registers the following AVP Codes for them in the "Authentication, Authorization, and Accounting (AAA) Parameters" registry:

<table>
<thead>
<tr>
<th>AVP Code</th>
<th>Attribute Name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD0</td>
<td>SASL-Mechanism</td>
<td>(this spec)</td>
</tr>
<tr>
<td>TBD1</td>
<td>SASL-Token</td>
<td>(this spec)</td>
</tr>
<tr>
<td>TBD2</td>
<td>SASL-Channel-Binding</td>
<td>(this spec)</td>
</tr>
</tbody>
</table>

7. References

7.1. Normative References


7.2. Informative References


Appendix A. Acknowledgements

Thanks go to TODO for useful discussions during the creation of this document.

Author’s Address

Rick van Rein
ARPA2.net
Haarlebrink 5
Enschede, Overijssel 7544 WP
The Netherlands

Email: rick@openfortress.nl