Workgroup: Network Working Group
Internet-Draft: draft-vanrein-diameter-sasl-05
Published: 6 August 2021
Intended Status: Informational
Expires: 7 February 2022
Authors: R. Van Rein
OpenFortress BV
H. Manson
Mansoft

Realm Crossover for SASL and GSS-API via Diameter

Abstract

SASL and GSS-API are used for authentication in many application protocols. This specification extends them to allow credentials of a home realm to be used against external services. To this end, it introduces end-to-end encryption for SASL that is safe to relay through a foreign server.

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 7 February 2022.

Table of Contents

1. Introduction

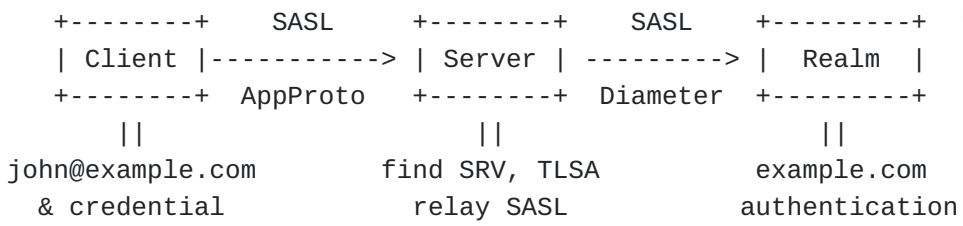
It is common for Internet users to work with services from a varierity of providers. An ad hoc practice has arisen of using local identity schemes for each of these providers. There is no integration of identity systems, and the practice reduces the control of users over their online identity. A solution to this is support for realm crossover TODO.xref.target=draft-vanrein-internetwide-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] and GSS-API [RFC2743] together 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 with the application protocol. SASL can easily

be funneled from one protocol into another, modulo a number of security concerns.

Diameter and its Network Access Server application are instrumental in authenticating a user under a realm, while not handing over any resources like an application protocol would. Furthermore, Diameter integrates with realm-crossing security; service can be declared under a domain name in a manner that is standardised, scalable and secure.

This can be used by a foreign server to authenticate a client with a backcall to the client's own domain:



Realm Crossover authentication:

Client John authenticates to his own Realm
while using a foreign Server.

The Diameter server in the Realm needs to respond success or failure on the SASL exchange forwarded to it. It delivers a User-Name on success, but not its domain. The client domain is validated by the foreign server, using DANE [RFC6698]. The combined User-Name and validated domain form the client identity as further used in the foreign server. The client realm also validates the foreign server, and MAY use this for access control, and perhaps to decide on the release of additional AVPs.

The client needs to assure that the authentication exchange cannot be relayed anywhere but to the Diameter service in his realm. This can be assured with channel binding [RFC5056] [RFC5801]; the foreign server detects this information and relays it to the Diameter service. No server accepts externally dictated channel binding information; the reason why it is safe to make an exception for Diameter is that it provides no resources, which makes it an unattractive attack target.

SASL mechanisms are not generally safe to pass over plaintext channels. This is usually addressed by wrapping the application protocol in TLS, but since that would only protect one leg of the intended realm-crossing authentication exchange, there is a need for end-to-end encryption.

This specification describes a SASL mechanism named SXOVER-PLUS as an end-to-end encrypted tunnel around another SASL exchange. It also defines how SASL can be embedded in a Diameter authentication exchange, which may be useful with SXOVER-PLUS or any other SASL mechanism.

Realm crossover for SASL is part of a series of protocol enhancements, as overviewed in TODO:xref target="draft-vanrein-internetwide-realm-crossover". Among the potential use cases are a global identity scheme for general communication and group participation, establishment of encryption keys, all with identity control under individually owned domains.

2. Messages of SXOVER-PLUS

SXOVER-PLUS consists of a few messages that develop an encryption secret and then continue using it as an end-to-end encrypted tunnel around a standard SASL authentication exchange. SXOVER continues to be active as long as the tunneled exchange does.

2.1. Preparation for Messaging

Before SXOVER-PLUS starts, the user submits a multi-session key to his realm and receives back a keyno and encalg in the style of Kerberos [RFC4120] along with a "keymap" blob that contain the originally submitted multi-session key. This process may be run at any time desired by the client; for instance, when a program first uses the SXOVER-PLUS mechanism; it may be kept for the remainder of the program run, even if this lasts for weeks and crosses between security realms, as a pre-validated key for protected contact with their realm; at any time, they can drop the key.

By offering the SXOVER-PLUS mechanism for SASL, a foreign server announces its willingness to validate the client's Realm as a domain, relay SASL messages to it, trust its authentication conclusion and User-Name and place it under the client's domain name.

Offering SXOVER-PLUS does not preclude the offering of other SASL mechanisms; for instance, ANONYMOUS may be useful to allow clients to choose guest access.

2.2. Initial Client-to-Server Message

SXOVER-PLUS is a client-first mechanism. The first SASL Token starts with "p=CHANBIND,,DOMAIN," where CHANBIND is the channel binding name and DOMAIN is the domain name of the client. This notation is compatible with the GS2 bridge [RFC5056].

Following this is DER-encoded information for the following ASN.1 structure:

```
C2S-Init ::= [APPLICATION 1] IMPLICIT SEQUENCE {
   clirnd   OCTET STRING,   -- Entropy to allow client variety
   keyno    KeyNumber,      -- With realm and encalg, identifies...
   encalg   EncryptAlg,     -- ...the key for keymap decryption...
   keymap   OCTET STRING    -- ...yielding server-acceptable data
}

EncryptAlg ::= Int32
KeyNumber  ::= UInt32
```

The clirnd is a salt that should hold enough entropy to satisfy the client's cryptographic requirements. The other fields result from the setup of the multi-session key preceding SXOVER-PLUS.

Upon reception, the server locates a key for the keyno and encalg in the key store for DOMAIN and uses it to decrypt keymap into entropy that serves as input to the random-to-key function [RFC3961], where the length of the decrypted keymap must match the key-generation seed-length.

The same key is constructed with random-to-key on both ends; the client uses the key that it originally submitted to the server. The result is now on both ends, and known as key K0.

Both ends pass K0 into the PRF+() function from RFC 6113 with the entire C2S-Init message (featuring the GS2 header and the entropy in the clirnd field) to produce properly sized input to the random-to-key function. The result is known as key K1. Note how this is similar to the KRB-FX-C2 procedure from RFC 6113, except that it is applied to a single key. (Considering slight generalisation of the procedure to a list of key/pepper pairs that are composed with associative/commutative XOR operators.)

2.3. Initial Server-to-Client Message

After the client-first SASL Token, the server sends its first challenge. It is encoded with DER and encrypted by K1, and contains the following ASN.1 structure:

```
S2C-Init ::= [APPLICATION 2] IMPLICIT SEQUENCE {
   srvrnd     OCTET STRING,   -- Entropy to allow server variety
   mechlist   IA5String       -- Available SASL mechanisms
}
```

The clirnd is a salt that should hold enough entropy to satisfy the client's cryptographic requirements. Note that the mechlist and DER tagging add no entropy.

The mechlist starts the SASL exchange inside the end-to-end encrypted tunnel. If this inner list uses channel binding at all, it should replicate the channel binding choices from the outer layer. Note that weaker channel binding choices such as tls-server-end-point may be met with a replay-protective mechlist.

The key K1 is passed into the PRF+() function from RFC 6113 with the pepper set to the concatenation of the entire S2C-Init message and the channel binding value. This is used to produce a last input to the random-to-key function. The result is known as key K2 and will be used to encrypt further messages, to be described as C2S-Cont and S2C-Cont.

The direct concatenation of S2C-Init with channel binding information is secure because of the self-descriptive size of the DER encoding of the former. Also note that there is no risk of cross-polination between types of channel binding because the name for the type has been hashed into key K1 and is therefore already securely encompassed in the key derivation.

2.4. Continued Client-to-Server Messages

Further messages from the client to the server hold DER content encrypted with key K2, following this ASN.1 format:

```
C2S-Cont ::= [APPLICATION 3] IMPLICIT SEQUENCE {
   mechsel   IA5String OPTIONAL,   -- SASL mechanism name selection
   c2s       SaslToken             -- NULL or SASL token passed
                                   -- from client to server
}

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

The mechsel indicates the client's choice of a SASL mechanism, and MUST be in the first inner SASL message. It initiates a new authentication exchange. The c2s holds the SASL Token and is sent as NULL whenever the mechanism yields no token, which is distinct from yielding an empty token.

The inner SASL exchange may be used to select an authorisation name that differs from the authentication name. This would be subject to normal approval by the SASL server, but upon success the authorisation name would be revealed in the User-Name over Diameter, and the foreign server would not be told about the authentication name. This can facilitate pseudonymity.

2.5. Continued Server-to-Client Messages

Further messages from the server to the client hold DER content encrypted with key K2, following this ASN.1 format:

```
S2C-Cont ::= [APPLICATION 4] IMPLICIT SEQUENCE {
   success  BOOLEAN DEFAULT FALSE,  -- When TRUE, s2c is an
                                    -- additional token
   s2c      SaslToken               -- NULL or SASL token from
                                    -- server to client
}
```

The s2c field carries the SASL Token if it is provided, even when it is empty, or it explicitly carries NULL to indicate an absent token. The success field may be set to TRUE to mark the provision of additional data upon success, and should be taken as a hint that no further SASL exchange is needed.

Note how this always facilitates last-sends by the SASL server. This is trivially done in Diameter, by just adding a SASL-Token AVP to the final success message; it is not always possible in the protocol between the client machine and foreign server, but that may be remedied by sending success in S2C-Cont and going through another looping to finish.

2.6. Using SXOVER-PLUS with GSS-API

When SXOVER-PLUS is used with GSS-API instead of SASL there are only a few changes to observe.

GSS-API Calls [RFC2744] to gss_init_sec_context() and gss_accept_sec_context() MUST adhere to [Section 5.1 of [RFC5801]] concerning channel binding informtion. Providing the GS2 header and channel binding data in the application-data field involves the "p=CHANBIND,," but not the "DOMAIN," part of the SASL header.

When transmitted as GSSAPI, only the first message changes. The client is now referred to as initiator and the server as acceptor.

In the first message, the initial part "p=CHANBIND,," is removed, but the "DOMAIN," and subsequent DER-encoded C2S-Init structure are kept. The standard GSSAPI header inserted in its place, adhering to the Mechanism-Independent Token Format [Section 3.1 of [RFC2743]] with object identifier 1.3.6.1.4.1.44469.666.5081.1 (TBD:GSSOID) to identify SXOVER-PLUS. When this object identifier is supplied to the call GSS_Inquire_SASLname_for_mech [Section 10 of [RFC5801]], the output reads "SXOVER-PLUS" (without the quotes).

TODO: Reconstruct SASL header or skip the "p=CHANBIND,," part in both SASL and GSS-API? Can we tell the channel binding type?

2.7. Application Key Derivation

SXOVER-PLUS adheres to most of the GS2 bridge, but deviates in two points. First, security layers are not considered useful in GS2 [Section 12 of [RFC5801]] because it assumes a secure layer that provides this benefit. With SXOVER-PLUS however, the end-to-end connection between a client and their authentication server differs from the single-hop connection to the foreign service, and it can be beneficial to extract secret key information between the former and latter. The second deviation from GS2 is that SXOVER-PLUS is defined but SXOVER is not. For these reasons, GS2- was not prefixed to the mechanism name.

In general, security layers may be derived from the key K2 by yet another pass through the PRF+() function from RFC 6113. The pepper for this is application-specific, and the requested length of octet-string can also be requested by the application. Multiple keys can be defined, each constructed from K2 and pepper.

Specifically, when SXOVER-PLUS is used under GSS-API, the following 32-byte ASCII strings may be used as pepper to derive keys for each of the four secure streams supported by GSS-API:

Pepper as 32 ASCII bytes	Purpose	Direction
SXOVER-PLUS/GSS-API/SIGN-C2S-KEY	signing	client --> server
SXOVER-PLUS/GSS-API/SIGN-S2C-KEY	signing	client <-- server
SXOVER-PLUS/GSS-API/WRAP-C2S-KEY	wrapping	client --> server
SXOVER-PLUS/GSS-API/WRAP-S2C-KEY	wrapping	client <-- server

Definitions for one application do not preclude the generation of keys for other applications. It is however vital to security that they all use different pepper, especially among different security contexts.

3. AVP Definitions for SASL in Diameter

SASL messages in Diameter use a number of AVPs [Section 4 of [RFC6733]] that are combined to relay SASL to an authentication realm.

These AVPs are added to the set that is used with the Network Access Server application [RFC7155], and can therefore be used in AA-Request and AA-Answer messages. They are always sent with the Mandatory Flag set to 0. When they are not recognised upon reception, they will be silently igored.

Normally, a successful AA-Answer would provide a User-Name AVP to inform the server about a username NAI without a realm [Section 2.1 of [RFC4282]] under which the client is identified; without the

User-Name an anonymous session is the only available option, possibly leading to reduced service and/or limited storage options. Sending a pseudonym in the User-Name may be an intermediate option. In all cases, the realm under which a successful AA-Answer is considered to fall can be taken from the Destination-Realm handling the Network Access Server session.

3.1. SASL-Mechanism

The SASL-Mechanism AVP has AVP Code TBD0 and is of type UTF8String, further restricted to a list of zero or more SASL mechanism names in their standardised notation [Section 3.1 of [RFC4422]] separated by a space character U+0020.

To retrieve a server's list of supported SASL mechanisms, this AVP is included in an AA-Request message, containing an empty list of SASL mechanism names, so an empty string. When SASL is supported by the server, it responds with the list of currently available SASL mechanisms.

To relay a client's choice of SASL mechanism, this AVP is included in an AA-Request message, containing a single SASL mechanism name. This MAY be done in another session than the one that retrieved the supported SASL mechanisms from the server, as long as origin and use have a matching Destination-Realm, because the SASL mechanism list has no other dependencies.

When the supported SASL mechanism list on a server is changed, any sessions that may count on one or more of the removed mechanisms SHOULD be aborted by the server. This is less likely to apply to client sessions that already selected a SASL mechanism. Clients MAY retrieve the server's supported mechanism list without actually attempting authentication; this can be a caching mechanism for a given Destination-Realm. An abort of such a session by the server indicates that the cache entry has expired, and should be retrieved anew for a following attempt.

3.2. SASL-Token

The SASL-Token AVP has AVP Code TBD1 and is of type OctetString. It may be passed in AA-Request and AA-Answer messages.

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.

The interpretation of a SASL-Token is subject to the SASL mechanism selection by the client. This is relayed with a SASL-Mechanism AVP,

which MUST be part of each Network Access Server session, no later than the first SASL-Token exchange.

3.3. SASL-Channel-Binding

The SASL-Channel-Binding AVP has AVP Code TBD2 and is of type OctetString. The AVP contains the literal channel binding information for a SASL mechanism, and may be sent in an AA-Request that also holds a SASL-Mechanism AVP that lists a single SASL mechanism.

Note that SASL requires channel binding information when the SASL-Mechanism AVP ends in -PLUS. Also note that different kinds of channel binding exist, and that they all start with a unique prefix registered with IANA. As a result, more than one SASL-Channel-Binding AVP can be included in one AA-Request. Servers MAY refrain from learning the client-chosen kind of channel binding from the SASL exchange, but SHOULD then transmit all the kinds that they support to avoid authentication failure.

4. Diameter Session Requirements for SASL

Probes for SASL mechanism lists SHOULD be sent outside of a Diameter session, and the response MAY be influenced by the Destination-Realm, Origin-Realm and Origin-Host AVPs. It SHOULD NOT be varied for other reasons.

Non-empty SASL-Mechanism AVPs, as well as any SASL-Token and SASL-Channel-Binding AVPs SHOULD NOT be sent outside of a Diameter session. The first AA-Request in this session SHOULD hold the SASL-Mechanism and MAY hold the SASL-Channel-Binding; these two AVPs SHOULD NOT occur in later messages in the same session. There MAY be a SASL-Token AVP in any AA-Request or AA-Answer anywhere in the Diameter session.

5. Diameter Message Requirements for SXOVER-PLUS

This section explains how the various SXOVER-PLUS messages are forwarded over Diameter by the foreign server. The foreign server is connected to the SASL client, possibly over a TLS connection or a protocol under GSS-API protection, and relays requests over Diameter, usually over SCTP with DTLS.

Diameter servers eventually provide success and failure responses, based on the corresponding final results from a SASL implementation that they in turn use. Before the final result is reached, the SASL implementation may impose a challenge that will be reproduced over Diameter, passing challenge and response tokens over Diameter on behalf of SASL.

5.1. C2S-Init Requests over Diameter

To send C2S-Init the Diameter client MUST include at least the following AVPs in an AA-Request [Section 3.1 of [RFC7155]]:

Destination-Realm is the client's requested realm, replicated here for Diameter routing purposes; SXOVER-PLUS conveys this value in plaintext;

SASL-Mechanism MUST be set to the fixed string SXOVER-PLUS for this SASL mechanism's name;

SASL-Token MUST be set to the C2S-Init and optional C2S-Cont as it produced by the SASL client;

SASL-Channel-Binding MUST be set to the channel binding bytes for the connection in which the SASL client attempts authentication, adhering to the channel binding mechanism named in the gs2-header in the SASL-Token.

It is possible to extend the message with more AVPs that the client and server can agree on.

The C2S-Init Request is likely to hold other Diameter AVPs for general housekeeping of the Diameter base protocol and NAS application, such as the Session-Id. Though User-Name and User-Password would be sent with password-based Diameter mechanisms, they MUST be ignored by implementations of SASL over Diameter when they appear in C2S-Init messages.

5.2. S2C-Init Responses over Diameter

When SASL fails to initialise in response to the C2S-Init passed in an AA-Request, then the AA-Answer MUST represent that in the following AVP:

Result-Code MUST be set to an error or failure code [Section 7.1 of [RFC6733]].

Upon initialisation of SASL, the normal response is a list of mechanisms that the client may use. If the AA-Request sent along a C2S-Cont that guessed an available mechanism and if that extension is acceptable to the server, then further processing will be as defined for S2C-Cont, below. Otherwise, the remainder of this section applies.

The initialisation of SASL forms a S2C-Init response, and an AA-Answer MUST be sent with the following AVPs:

Result-Code

MUST be set to the value DIAMETER_MULTI_ROUND_AUTH [Section 7.1.1 of [RFC6733]];

SASL-Token MUST be set to the S2C-Init value.

5.3. C2S-Cont Requests over Diameter

The C2S-Cont message is any further message that the SASL client passes to the foreign server. It MUST be forwarded as a Diameter AA-Request with the following AVPs:

SASL-Token MUST be set to the C2S-Cont value from the SASL client;

SASL-Mechanism MUST NOT be sent;

SASL-Channel-Binding MUST NOT be sent;

User-Name MAY be sent but MUST NOT be processed when received by implementations of this specification;

User-Password MOST NOT be sent.

5.4. S2C-Cont Responses over Diameter

S2C-Cont tokens are produced as output from continued SASL processing based on C2S-Cont tokens found in AA-Request messages.

If the SASL exchange is not final, then the AA-Answer MUST represent that in the following AVPs:

Result-Code is set to the value DIAMETER_MULTI_ROUND_AUTH [Section 7.1.1 of [RFC6733]];

SASL-Token MUST be included, and set to the S2C-Cont value; when responding to accepted optimisation for the initial round-trip then the S2C-Init token MUST be prefixed to the S2C-Cont value.

If the SASL exchange fails, then the AA-Answer MUST represent that in the following AVP:

Result-Code is set to an error or failure code [Section 7.1 of [RFC6733]].

If the SASL exchange succeeds, then the AA-Answer MUST represent that in the following AVPs:

Result-Code is set to a success code [Section 7.1.2 of [RFC6733]];

SASL-Token is included when the SASL exchange produced an additional token upon success [Section 4 of [RFC4422]];

User-Name

may be provided, and then contains the username part of a NAI [RFC4282], but not a realm; when an authorization identity string was provided [Section 3.4.1 of [RFC4422]] and approved by the SASL exchange, then this will be used instead of the authentication idenity. This mechanism may be used to request the use of a pseudonym as well as to signal the willingness to return this AVP.

Further AVPs may be included in a successful AA-Answer, but their meaning is not defined herein. Applications might range from access control lists to backend tunnel creation.

6. Running Diameter as a SASL Backend

Following are a few practical considerations in relation to the Diameter connectivity for SASL.

6.1. Diameter is an SCTP service

Diameter is primarily an SCTP-based protocol [RFC6733], for reasons of scalabaility and efficiency. SASL Diameter benefits from these properties and embraces the SCTP transport. 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 then sacrifices several benefits of the SCTP carrier. SASL Diameter embeddings typically involve no client systems, so 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.

6.2. 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 MAY choose to raise the bar by setting a minimal accepable level.

DNSSEC offers a 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, can be used as specified for DANE [RFC6698]. This use of DNSSEC and DANE is RECOMMENDED.

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.

7. Security Considerations

The SASL mechanism SXOVER-PLUS separates the authentication of a foreign identity into its realm and the username underneath it. The realm is authenticated by the relying server, such as the proposed foreign server, whereas the username is obtained from a backend realm server that is known to be responsible for that realm.

From the perspective of the foreign server, assurance of an identity is the vital aspect of the SXOVER-PLUS flow that it relays over Diameter. Through TLS or DTLS, with DNSSEC and DANE to validate the certificate it uses, the link from a realm (which is read as a domain name) to the Diameter connection can be verified, so the relying server can be certain about the realm under which the backend connection resides. By receiving a response over that connection to a known-authoritative server for the realm, the username can also be trusted. The relying server continues to treat the username and realm as a pair the for identification of the user.

Channel binding is normally limited to two parties only, and forwarding such information is not a trivial idea. The fact that the forwarding connection is encrypted, and known to lead to an authoritative server for a claimed realm does help. The intermediate server relies on proper authentication, and has no interest in bypassing authentication, and it would be doing that by adopting channel binding information from anywhere else.

From the perspective of the client and the home realm, the safety of the SASL credentials is paramount. When addressing a foreign server, which is not part of the home realm, clients therefore MUST NOT rely on mechanisms that might leak credentials. Two mechanisms that are safe to use are ANONYMOUS, which passes no credentials and assigns no rights, and SXOVER-PLUS, which applies end-to-end encryption to another SASL mechanism that may or may not be secure.

The SXOVER-PLUS mechanism uses channel binding to ensure that the authentication is specific to a stream. The level to which this is secure depends on the channel binding mechanism. Therefore, in spite of end-to-end encryption, most use cases will want a secure carrier such as TLS between the client and foreign server.

8. IANA Considerations

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

AVP Code	Attribute Name	Reference
TBD0	SASL-Mechanism	(this spec)
TBD1	SASL-Token	(this spec)
TBD2	SASL-Channel-Binding	(this spec)

This specification defines a new value for the NAS-Port-Type AVP to indicate a new interpretation of values passed in NAS-Port and NAS-Port-Id AVPs. IANA is requested to register the following value in the RADIUS Types registry, under Values for RADIUS Attribute 61, NAS-Port-Type:

Value	Description	Reference
TBD3	SASL Authenticated Service	(this spec)

This specification defines a SASL mechanism named SXOVER-PLUS. IANA is requested to register the following in the Simple Authentication and Security Layer (SASL) Mechanisms registry under SASL Mechanisms:

Mechanism	Usage	Reference	Owner
SXOVER-PLUS	COMMON	(this spec)	Rick van Rein <rick@openfortress.nl

9. Normative References

[I-D.vanrein-internetwide-realm-crossover] Rein, R. V., "InternetWide Identities with Realm Crossover", Work in Progress, Internet-Draft, draft-vanrein-internetwide-realm-crossover-00, 28 September 2020, <https://www.ietf.org/archive/id/draft-vanrein-internetwide-realm-crossover-00.txt>.

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, DOI 10.17487/RFC2743, January 2000, <https://www.rfc-editor.org/info/rfc2743>.

[RFC2744] Wray, J., "Generic Security Service API Version 2 : C-bindings", RFC 2744, DOI 10.17487/RFC2744, January 2000, <https://www.rfc-editor.org/info/rfc2744>.

[RFC3961] Raeburn, K., "Encryption and Checksum Specifications for Kerberos 5", RFC 3961, DOI 10.17487/RFC3961, February 2005, <https://www.rfc-editor.org/info/rfc3961>.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, DOI 10.17487/RFC4120, July 2005, <https://www.rfc-editor.org/info/rfc4120>.

[RFC4282] Aboba, B., Beadles, M., Arkko, J., and P. Eronen, "The Network Access Identifier", RFC 4282, DOI 10.17487/RFC4282, December 2005, <https://www.rfc-editor.org/info/rfc4282>.

[RFC4422] Melnikov, A., Ed. and K. Zeilenga, Ed., "Simple Authentication and Security Layer (SASL)", RFC 4422, DOI 10.17487/RFC4422, June 2006, <https://www.rfc-editor.org/info/rfc4422>.

[RFC5056] Williams, N., "On the Use of Channel Bindings to Secure Channels", RFC 5056, DOI 10.17487/RFC5056, November 2007, <https://www.rfc-editor.org/info/rfc5056>.

[RFC5554] Williams, N., "Clarifications and Extensions to the Generic Security Service Application Program Interface (GSS-API) for the Use of Channel Bindings", RFC 5554, DOI 10.17487/RFC5554, May 2009, <https://www.rfc-editor.org/info/rfc5554>.

[RFC5801] Josefsson, S. and N. Williams, "Using Generic Security Service Application Program Interface (GSS-API) Mechanisms in Simple Authentication and Security Layer (SASL): The GS2 Mechanism Family", RFC 5801, DOI 10.17487/RFC5801, July 2010, <https://www.rfc-editor.org/info/rfc5801>.

[RFC5929] Altman, J., Williams, N., and L. Zhu, "Channel Bindings for TLS", RFC 5929, DOI 10.17487/RFC5929, July 2010, <https://www.rfc-editor.org/info/rfc5929>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <https://www.rfc-editor.org/info/rfc6698>.

[RFC6733] Fajardo, V., Ed., Arkko, J., Loughney, J., and G. Zorn, Ed., "Diameter Base Protocol", RFC 6733, DOI 10.17487/RFC6733, October 2012, <https://www.rfc-editor.org/info/rfc6733>.

[RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, DOI 10.17487/RFC6951, May 2013, <https://www.rfc-editor.org/info/rfc6951>.

[RFC7155] Zorn, G., Ed., "Diameter Network Access Server Application", RFC 7155, DOI 10.17487/RFC7155, April 2014, <https://www.rfc-editor.org/info/rfc7155>.

Appendix A. Centralised handing of SASL over Diameter

This section is non-normative.

Within foreign service networks, it can be usefule to centralise Diameter handling in one peer, where service-neutral pooling of connections to remote peers can improve efficiency. Diameter could facilitate this, but would add quite a bit of handling logic to a foreign service. The following ASN.1 module was designed as the simplest possible request/response protocol that could run over a TCP connection between a foreign service host and a nearby/trusted centralised host running its side of Diameter.

The protocol can also be used over SCTP. In this case, a user message can be defined to contain precisely one DiaSASL-Request in downstream direction, or one DiaSASL-Answer in upstream direction, and sent with the SCTP_UNORDERED flag.

```
Quick-DiaSASL DEFINITIONS EXPLICIT TAGS ::= BEGIN

-- ## SASL ready for Diameter
--
-- This is targeted at Diameter backends and avoids loading all of
-- Diameter into applications.
--

-- Open a connection; return is DiaSASL-Open-Answer.
-- The service-realm defines the context of the
-- identity provider; this is where Diameter requests
-- should be send, and it helps to determine what
-- sasl-mechanisms may be used.
--
-- The front-end is identified by a service-trunk code
-- (for the long-term relation between a front-end and
-- back-end) and/or a service-proto protocol that can
-- be used while driving SASL (it could be the "imap"
-- part before the "imap/imap.example.com"PrincipalName
-- for a service in a Kerberos Ticket).
--
DiaSASL-Open-Request ::= [APPLICATION 10] IMPLICIT SEQUENCE {
   service-realm   [1] UTF8String,
   service-trunk   [8] INTEGER   OPTIONAL,
   service-proto   [9] IA5String OPTIONAL
}

-- Close a connection; session-id identifies which
-- and there is no response.  This is ignored when the
-- session-id is unknown; the call is not required
-- after a DiaSASL-Authn-Answer that sets a value for
-- final-comerr, but it is harmless when sent anyway.
--
DiaSASL-Close-Request ::= [APPLICATION 11] IMPLICIT SEQUENCE {
   session-id   [2] OCTET STRING
}

-- Relay an authentication request message; response is
-- DiaSASL-Authn-Answer with a copied session-id.
--
DiaSASL-Authn-Request ::= [APPLICATION 12] IMPLICIT SEQUENCE {
   session-id             [2] OCTET STRING,
   sasl-mechanism         [3] IA5String OPTIONAL,
   sasl-channel-binding   [4] OCTET STRING OPTIONAL,
   sasl-token             [5] OCTET STRING OPTIONAL
}

-- This is the response to a DiaSASL-Open-Request.
--
```

```
-- The final-comerr is set when Diameter was conclusive.
-- It is an error code from com_err to allow for errors,
-- but it may be sufficient to know that 0 indicates success
-- and everything else is a failure.
--
-- The service-realm is copied from the Diasasl-Open-Request
-- so it can be used to match; the session-id will continue
-- to identify this session in requests and responses.
--
-- The sasl-mechanisms holds a space-separated string of
-- SASL mechanism names.
--
DiaSASL-Open-Answer ::= [APPLICATION 13] IMPLICIT SEQUENCE {
   final-comerr      [0] INTEGER (-2147483648..2147483647) OPTIONAL,
                         -- Only set when Diameter was conclusive.
                         -- 0 for success, different for failure.
                         -- The code is a com_err code, so int32_t.
   service-realm     [1] UTF8String,
   session-id        [2] OCTET STRING,
   sasl-mechanisms   [3] IA5String
}

-- This is the response to a DiaSASL-Authn-Request.
--
-- The final-result is only set if Diameter was conclusive.
-- It is an error code from com_err to allow for errors,
-- but it may be sufficient to know that 0 indicates success
-- and everything else is a failure.
--
-- Only a successful authentication response can hold values
-- for client-userid and client-domain.  The latter overrides
-- the initial realm, which was provided in the open call,
-- but may be substituted as a result of Realm Crossover.
-- The client-userid is the local part and may be absent on
-- anonymous sessions; the client-userid value is approved
-- by the local Diameter peer as having come from a Diameter
-- Diameter peer that tends to client-domain.
--
DiaSASL-Authn-Answer ::= [APPLICATION 14] IMPLICIT SEQUENCE {
   final-comerr   [0] INTEGER (-2147483648..2147483647) OPTIONAL,
                      -- Only set when Diameter was conclusive.
                      -- 0 for success, different for failure.
                      -- The code is a com_err code, so int32_t.
   session-id     [2] OCTET STRING,
   sasl-token     [5] OCTET STRING OPTIONAL,
   client-userid  [6] UTF8String OPTIONAL,
   client-domain  [7] UTF8String OPTIONAL
}
```

```
-- Requests are Open, Close and Authn requests.  This simple
-- CHOICE differentiates between the variants.
-- Note that no extra tags are needed; the [APPLICATION n]
-- tag can be used, or the presence of fields in variants.
--
DiaSASL-Request ::= CHOICE {
    open-request   DiaSASL-Open-Request,
    close-request  DiaSASL-Close-Request,
    authn-request  DiaSASL-Authn-Request
}

-- Answers are sent in response to Open and Authn requests.
-- This simple CHOICE differentiates between the variants.
-- Note that no extra tags are needed; the [APPLICATION n]
-- tag can be used, or the presence of fields in variants.
--
DiaSASL-Answer ::= CHOICE {
    open-answer    DiaSASL-Open-Answer,
    authn-answer   DiaSASL-Authn-Answer
}


-- ## A simple API for DiaSASL

-- A `diasasl` API only needs a small number of calls:
-- http://quick-sasl.arpa2.net/group__quickdiasasl.html
-- This presents only a modest extension to existing software,
-- and easily merges into a variety of concurrency models.

END
```

Appendix B. Acknowledgements

Thanks to Nico Williams for input on the GS2 bridge and Channel Binding.

Authors' Addresses

Rick van Rein
OpenFortress BV
Haarlebrink 5
Enschede

Email: rick@openfortress.nl

Henri Manson
Mansoft
Castorstraat 30
Enschede

Email: info@mansoft.nl