Realm Crossover for SASL and GSS-API via Diameter
draft-vanrein-diameter-sasl-03

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 to
the client’s home realm.

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 July 24, 2020.

Copyright Notice

Copyright (c) 2020 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 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 notifications. 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-crossing authentication exchange, there is 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. Messages of GS2-SXOVER-PLUS

GS2-SXOVER-PLUS establishes a session key and continues with an encrypted, but otherwise normal SASL exchange. This cloak provides end-to-end encryption for the contained SASL exchange, which allows it to be passed through a foreign server. This section defines the messages involved in the GS2-SXOVER-PLUS exchange in isolation, later sections specify how this fits into GSS-API and SASL contexts, and how these travel to the home realm’s network access server.
By announcing the GS2-SXOVER-PLUS mechanism for SASL, a foreign server declares it is willing to relay SASL messages for that mechanism to the authentication realm indicated by the client in its first GS2-SXOVER-PLUS message in a plaintext form. Later sections define a standard mechanism for relaying such messages over Diameter, using DNSSEC and DANE so that the result can be trusted to have come from the indicated realm and thus warrant any user name scoped under that realm. Offering GS2-SXOVER-PLUS does not preclude the offering of other SASL mechanisms; for instance, ANONYMOUS may be useful to allow clients to choose guest access.

2.1. Initial Client-to-Server Message

The GS2-SXOVER-PLUS exchange is initiated by the client, by sending a C2S-Init message:

C2S-Init ::= [APPLICATION 1] IMPLICIT SEQUENCE {
  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 inictr value is used as a bit of entropy, and will be incremented by one for every next message in the same flow. This helps to bind messages into one flow and to distinguish among flows. Though helpful to protect against replay attacks, dynamic challenges and channel binding offer better protection.

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 method by which the keyno and encalg and the key itself are established is not defined here, because its choice is local to the client’s realm. The reason for not authenticating with this key is that anonymous encryption keys are much easier to establish than authentication keys. One possible mechanism is KIP, the Keyful Identity Protocol, but it is not prescribed herein.

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. Initial Server-to-Client Message

In both SASL and GSS-API, this first message from server to client is
passed as an S2C-Init message:

S2C-Init ::= [APPLICATION 2] IMPLICIT SEQUENCE {
    ctr        Counter,              -- Counter value is inictr+1
    realm      IA5String,            -- Secure realm confirmation
    mechlist   SEQUENCE OF IA5String -- Available SASL mechanisms
}

The first message from the server to the client is named the Initial
Response in SASL. This message presents the client with a choice of
mechanism names to use under the cloak of GS2-SXOVER-PLUS. In
addition, it securely confirms the realm name that will be assumed in
the upcoming exchange.

GS2-SXOVER-PLUS makes no assumptions about the mechanisms supported
at the home realm. Instead, S2C-Init lists mechanisms specific to
the home realm of the client, which are concealed from the foreign
server, so it cannot influence the list.

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 counter mechanism is both a protection against
message resending and a means of having concurrent SASL exchanges if
the client wants to.

The realm is repeated from the GS2-SXOVER-PLUS 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, GS2-
SXOVER-PLUS and ANONYMOUS mechanisms should not occur in the wrapped
mechlist. Furthermore, only mechanisms supporting channel binding
SHOULD be supported, meaning that all strings in mechlist should have
a -PLUS ending.
2.3. Continued Client-to-Server Messages

Further GS2-SXOVER-PLUS messages from client to server are named (initial) responses by SASL, and they are formed by encrypting the DER-form of C2S-Cont under the seskey. There is one special case, namely the need for the first C2S-Cont to select a SASL mechanism to run under the seskey cloak. For all C2S-Cont messages, there is a separate representation for no data, distinguishable from empty data:

C2S-Cont ::= [APPLICATION 3] IMPLICIT 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. The mechsel MUST support channel binding, so it must have a -PLUS ending. In any message, the ctr value is one more than the value in the previous SASL message, reduced to an unsigned 32-bit range; see Section 3.2 for an important detail caused by round-trip optimisation.

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 by SASL [RFC4422].

2.4. Continued Server-to-Client Messages

After the first message from the server to the client, they adhere to the structure of S2C-Cont, defined below. The SASL term for these messages would be a Challenge that is not an Initial Challenge. The exchange is encrypted under the seskey:

S2C-Cont ::= [APPLICATION 4] IMPLICIT 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 GS2-SXOVER-PLUS exchange, as it is part of the SASL mechanism that was selected with mechsel in the wrapped exchange. GS2-SXOVER-PLUS does not specify an extra value, so the field in the outer SASL exchange that runs GS2-SXOVER-PLUS will not be used.

3. GS2-SXOVER-PLUS as a GS2 Mechanism

The messages of GS2-SXOVER-PLUS will now be mapped to SASL and GSS-API. The GS2 bridge [RFC5056] defines the requirements to make the two behave equivalently on the basis of a header preceding the first "raw" message which is the C2S-Init message.

Further messages, so S2C-Init, C2S-Cont and S2C-Cont will be encrypted but otherwise used as-is under both SASL and GSS-API.

3.1. Encrypting GS2-SXOVER-PLUS Messages

Before inclusion in the GSS-API and SASL frames, most GS2-SXOVER-PLUS messages will be encrypted:

C2S-Init is not encrypted, but it contains a seskey field that is encrypted under the long-term key with key usage value KIP_KEYUSAGE_MAPPING or 1864;

S2C-Init, C2S-Cont, S2C-Cont are encrypted under the seskey, as supplied in the C2S-Init that started the session, with key usage value KIP_KEYUSAGE_USERDATA or 1863.

The encrypt operation [Section 3 of [RFC3961]] uses default initial state and is known to also guarantee integrity. The name KIP references the compatible Keyful Identity Protocol, which may or may not develop independently of GS2-SXOVER-PLUS.
3.2. Initial Round-trip Optimisation

This section introduces an optimisation that MUST be accepted by servers, and MAY be sent by clients. In addition, the server MAY use it in response to optimised messages to clients which MUST then accept it.

Normally, the C2S-Init, S2C-Init, C2S-Cont and S2C-Cont messages are all placed in a single GSS-API or SASL message, with encryption and headers applied as dictated for each occasion. The optimisation described here concatenates an opportunistic C2S-Cont to C2S-Init, and upon acceptance of the opportunistic attempt the server sends not only a S2C-Init, but also the accepting S2C-Cont.

Encryption is applied to the GS2-SXOVER-PLUS messages before they are concatenated. The header is applied after concatenation.

The Counter values used in case of success or failure of the opportunistic attempt are:

<table>
<thead>
<tr>
<th>Message</th>
<th>Counter on success</th>
<th>Counter on failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2S-Init</td>
<td>incr_t</td>
<td>incr_t</td>
</tr>
<tr>
<td>C2S-Cont</td>
<td>incr_t + 2</td>
<td>incr_t + 2</td>
</tr>
<tr>
<td>S2C-Init</td>
<td>incr_t + 1</td>
<td>incr_t + 1</td>
</tr>
<tr>
<td>S2C-Cont</td>
<td>incr_t + 3</td>
<td>(message absent)</td>
</tr>
<tr>
<td>next...</td>
<td>incr_t + 4,5,6...</td>
<td>incr_t + 3,4,5...</td>
</tr>
</tbody>
</table>

or, in words, the counter is not incremented when failure causes the S2C-Init to not be sent. This is reflected in the next messages, which would start with another attempted C2S-Cont, and normal counting commences from that point.

The use of this pattern is that a mechanism may be tried immediately, without awaiting the mechanism list. Very often, clients will be setup to validate using a particular mechanism, or they may have learnt from a prior exchange. This is particularly useful because traffic concentrates at the home realm, which usually leads to a stable mechanism list.

3.3. Initial Header for GSS-API

The header to use for GSS-API is standardised as a Mechanism-Independent Token Format [Section 3.1 of [RFC2743]] and prefixed to the initial token of a GSS-API context establishment sequence, incorporating the object identifier 1.3.6.1.4.1.44469.666.5081.1 (TBD:GSSOID) to identify GS2-SXOVER-PLUS. When this object identifier is supplied to the call GSS_Inquire_SASLname_for_mech
Following the header is a GSS-API variant of C2S-Init, which prefixes an authorization identity, interpreted as defined in the GS2 header for SASL. The structure after the header in ASN.1 notation is:

GSS-SXOVER-Init ::= SEQUENCE {
    gs2-authzid IA5String,
    c2s-init C2S-Init
}

The annotated bytes are shown below:

60 02 ...length... -- [APPLICATION 0] IMPLICIT SEQUENCE { ... }
   -- OBJECT IDENTIFIER 1.3.6.1.4.1.44469.666.5081.1 (TBD:GSSOID)
 06 ...length... 2b 06 01 04 01 44469 TBD:666 TBD:5081 01 -- OBJECT IDENTIFIER
   -- GSS-SXOVER-Init
 30 ...length... -- SEQUENCE { ... }
    ...

The mapping of GS2-SXOVER-PLUS into GSS-API is less natural than into SASL, because it references SASL mechanisms. This is not a strict problem however, and implementers MAY provide GS2-SXOVER-PLUS as a GSS-API mechanism. The absence of a realm name in the generic parts of the protocol may however lead to difficulties routing.

The function GSS_Inquire_SASLname_for_mech [Section 10 of [RFC5801]] maps the aforementioned object identifier for the GSS-API mechanism to the name "GS2-SXOVER-PLUS" (without quotes).

3.4. Initial Header for SASL

The header to use for SASL is the gs2-header [RFC5801] with a few extra constraints: constrained than its general form, namely:

gs2-nonstd-flag is absent;

gs2-cb-flag MUST NOT be "y" or "n" because channel binding is required; the only remaining general form is therefore ("p=" cb-name); a foreign server MUST interpret this flag and relay the appropriate channel binding information through its Diameter backend;

gs2-authzid is used for routing of C2S-Init to the Diameter server of the authoritative backend realm. This field MUST contain at least one AT symbol U+0040. The domain name for the backend is mentioned after the last AT symbol. Anything preceding the AT
symbol is interpreted by the local realm, and MAY be used for
realm-internal routing to a more specific Diameter backend
service node, but identification is done within the realm by
the inner SASL layer, and the foreign server MUST NOT rely on
text before the last AT symbol.

As an example, the gs2-header targeting a realm "example.com" and
channel binding through tls-unique could be

p=tls-unique,@example.com,

or, when example.com uses internal GS2-SXOVER-PLUS routing to a
service node named +idp+sxover it might be

p=tls-unique,+idp+sxover@example.com,

In both cases, the last comma would be immediately followed by the
DER-encoded S2C-Init structure. Since SASL tokens are always carried
over binary channels, there is no use in continuing the message in a
textual form.

4. 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. On top of that, the SASL-Mechanism AVP may
also occur in a Capabilities Exchange Answer. The User-Name AVP MUST
be supplied in a successful 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.

4.1. SASL-Mechanism

The SASL-Mechanism AVP has AVP Code TBD0. This specification only
uses the mechanism name GS2-SXOVER-PLUS as a value for this AVP. It
MUST be included in the first message of an GS2-SXOVER-PLUS 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.
4.2. SASL-Token

The SASL-Token AVP has AVP Code TBD1. 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.

4.3. SASL-Channel-Binding

The SASL-Channel-Binding AVP has AVP Code TBD2. It MUST appear along the first SASL-Token AVP for a Network Access session if the SASL-Mechanism ends in -PLUS.

This AVP may occur more than once, to indicate support of multiple forms of channel binding. Note however that all mechanisms suitable for Diameter relaying use the GS2 bridge [RFC5056] in which case the channel binding name to pass along in this message can be derived.

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 concatenates two values:

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 a SASL 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.

5. Diameter Message Requirements for GS2-SXOVER-PLUS

This section explains how the various GS2-SXOVER-PLUS messages are forwarded over Diameter by the foreign server. The foreign server is connected to the SASL client, usually over a TLS connection, and relays requests over Diameter, usually over SCTP with DTLS.

Diameter servers provide success and failure responses, based on the corresponding final results from a SASL service that they in turn use. When no such final result comes from a Diameter request, a
challenge will instead be produced over Diameter, holding a SASL challenge token from the server.

5.1. C2S-Init Requests over Diameter

To send C2S-Init, possibly including the C2S-Cont that the optimisation adds, the Diameter client MUST include at least the following AVPs in an AA-Request [Section 3.1 of [RFC7155]]:

Realm is the client’s requested realm, replicated here for routing purposes; GS2-SXOVER-PLUS provides this value in the gs2-header’s authorization identity field;

SASL-Mechanism is set to the fixed string GS2-SXOVER-PLUS;

SASL-Token is set to the C2S-Init and optional C2S-Cont as it arrived from the SASL client;

SASL-Channel-Binding is set to the channel binding information 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 if the client and server have agreed on this, perhaps as a result of capability negotiation during Diameter connection setup.

The C2S-Init Request is likely to hold other Diameter AVPs for general housekeeping of Diameter in general and the NAS application, such as Session-Id. Though User-Name and User-Password would be sent with password-based Diameter mechanisms, they are not required in C2S-Init messages, but they MAY be sent with empty contents to accommodate software and the RECOMMENDED status of the AVPs in the AA-Request, in which case they MUST be ignored on reception.

5.2. S2C-Init Responses over Diameter

The Diameter server serves as a SASL server to which the foreign server relays requests. To this end, the Diameter server responds with acceptance, denial or a further challenge. Acceptance and denial are used for the corresponding SASL outcomes, the further challenge also matches logically.

When the SASL server responds with S2C-Init over Diameter, the Diameter server MUST include at least the following AVPs in a successful AA-Answer [Section 3.2. of [RFC7155]]:
User-Name is set to the identity of the SASL client, which resulted from the encapsulated SASL exchange and possibly further authorisation processing in the SASL server; the name MUST NOT add the realm name in this attribute; it is up to the foreign server to assign it under the authoritative realm, possibly by appending the last U+0040 (AT) character and its realm name observed from the GS2 header sent in C2S-Init over Diameter.

Permissions may be expected from a Diameter server, but are considered optional for SASL over Diameter. The reason is that SASL focusses on authentication, not authorisation. This especially applies to realm crossover, where authentication is a matter of the home realm and authorisation is, at least by default, the prerogative of the foreign server implementing its own resource with its own semantics. Extensions to this specification could however be made to use the infrastructure proposed herein to also centralise the storage and/or processing of resource access rights.

If the SASL exchange requires continuation, then the AA-Answer represents a challenge to follow up, represented in the an AA-Answer that MUST include at least the following AVPs:

- **Result-Code** is set to the value DIAMETER_MULTI_ROUND_AUTH;
- **SASL-Token** is set to the S2C-Init value;
- **State** is set to server state to be reproduced in the followup.

If the AA-Answer is a final failure report, this MUST be represented in a failing Result-Code AVP.

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 is forwarded as a Diameter AA-Request [Section 3.1 of [RFC7155]] which MUST contain at least the following AVP:

- **SASL-Token** is set to the token from the SASL client.

The C2S-Cont Request MUST NOT contain AVPs for SASL-Mechanism or SASL-Channel-Binding. It is however likely to hold other Diameter AVPs for general housekeeping of Diameter in general and the NAS application, such as Session-Id and State AVPs. Though User-Password would be sent with password-based mechanisms, it is not required in C2S-Cont messages, but it MAY be sent with empty contents to
accommodate software and the RECOMMENDED status of the AVP, in which case it MUST be ignored on reception.

5.4. S2C-Cont Responses over Diameter

The S2C-Cont Response message may inform the Diameter client of success, failure or a further challenge. It is transmitted over Diameter as a Diameter AA-Answer [Section 3.2 of [RFC7155]] message, with the customary Result-Code interpretations.

Processing of these Diameter messages is the same as for S2C-Init, with the exception that the SASL-Token, if it is present, is not interpreted as S2C-Init but as S2C-Cont.

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 scalability and efficiency. SASL Diameter benefits from these properties and embraces the SCTP carrier. Operating system support for SCTP is widespread, 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 can choose to raise the bar by setting a minimum standard.

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 GS2-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 GS2-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 GS2-SXOVER-PLUS, which applies end-to-end encryption to another SASL mechanism that may or may not be secure.

The GS2-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 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>

9. Normative References


Appendix A.  Acknowledgements

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

Author’s Address

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

Email: rick@openfortress.nl