Network Working Group L. Howard
Internet-Draft PADL
Intended status: Informational April 20, 2020
Expires: October 22, 2020

A Simple Anonymous GSS-API Mechanism
draft-howard-gss-sanon-11

Abstract

This document defines protocols, procedures and conventions for a Generic Security Service Application Program Interface (GSS-API) security mechanism that provides key agreement without authentication of either party.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 2
2. Requirements notation 2
3. Discovery and Negotiation 3
4. Naming 3
4.1. Name Types 3
4.2. Canonicalization 4
4.3. Exported Name Format 4
5. Definitions and Token Formats 4
5.1. Context Establishment Tokens 4
5.1.1. Initial context token 4
5.1.2. Acceptor context token 5
5.1.3. Initiator context completion 6
5.2. Per-Message Tokens 6
5.3. Context Deletion Tokens 6
6. Key derivation 6
7. Pseudo-Random Function 7
8. Security Considerations 7
9. Acknowledgements 8
10. References 8
10.1. Normative References 8
10.2. Informative References 9
Appendix A. Test Vectors 9
Appendix B. Mechanism Attributes 10
Appendix C. NegoEx 10
Author's Address 11

1. Introduction

The Generic Security Service Application Program Interface (GSS-API) [RFC2743] provides a framework for authentication and message protection services through a common programming interface.

The Simple Anonymous mechanism (hereafter SAnon) described in this document is a simple protocol based on the X25519 elliptic curve Diffie-Hellman (ECDH) key agreement scheme defined in [RFC7748]. No authentication of initiator or acceptor is provided. A potential use of SAnon is to provide a degree of privacy when bootstrapping unkeyed entities.

2. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Discovery and Negotiation

The SAnon mechanism is identified by the following OID:

```
sanon-x25519 OBJECT IDENTIFIER ::=
    {iso(1)identified-organization(3)dod(6)internet(1)
     private(4)enterprise(1)padl(5322)gss-sanon(26)
     mechanisms(1)sanon-x25519(110)}
```

The means of discovering GSS-API peers and their supported mechanisms is out of this specification's scope. To avoid multiple layers of negotiation, SAnon is not crypto-agile; a future variant using a different algorithm would be assigned a different OID.

If anonymity is not desired then SAnon MUST NOT be used. Either party can test for the presence of GSS_C_ANON_FLAG to check if anonymous authentication was performed.

4. Naming

4.1. Name Types

The SAnon mechanism can import a variety of name types. A SAnon mechanism name is logically a boolean indicating whether it represents an anonymous identity. The following table indicates which names represent anonymous identities (the GSS_C_NT_ prefix is omitted for space):

Name type	Name string	AI
USER_NAME	WELLKNOWN/ANONYMOUS@WELLKNOWN:ANONYMOUS	Y
HOSTBASED_SERVICE	WELLKNOWN@ANONYMOUS	Y
ANONYMOUS	Any name string	Y
Any other name type	Any name string	N

Inferring an anonymous identity from a well known name permits applications that do not support GSS_C_ANON_FLAG or GSS_C_NT_ANONYMOUS, but which permit user-entered names, to use SAnon. However, as noted in [RFC8062], portable initiators are RECOMMENDED to use default credentials whenever possible and request anonymity only through the input anon_req_flag [RFC2743] to GSS_Init_sec_context().

4.2. Canonicalization

The canonical form of a SAnon mechanism name is the boolean indicating whether it represents an anonymous identity. When GSS_Display_name() is called on an anonymous mechanism name, the display string is WELLKNOWN/ANONYMOUS@WELLKNOWN:ANONYMOUS [RFC8062] and the name type is GSS_C_NT_ANONYMOUS. This anonymous identity is always the name observed by a SAnon peer. All context APIs that return peer names MUST return this name for both parties if the context is established.

4.3. Exported Name Format

SAnon uses the mechanism-independent exported name object format defined in [RFC2743] Section 3.2. All lengths are encoded as big-endian integers. The export of non-anonymous mechanism names MUST fail with GSS_S_BAD_NAME.

Length	Name	Description
2	TOK_ID	04 01
2	MECH_OID_LEN	Length of the mechanism OID
MECH_OID_LEN	MECH_OID	The SAnon mechanism OID, in DER
4	NAME_LEN	00 00 00 01
1	NAME	01

5. Definitions and Token Formats

5.1. Context Establishment Tokens

5.1.1. Initial context token

The initial context token is framed per Section 1 of [RFC2743]:

```
GSS-API DEFINITIONS ::=
    BEGIN

    MechType ::= OBJECT IDENTIFIER -- 1.3.6.1.4.1.5322.26.1.110
    GSSAPI-Token ::=
    [APPLICATION 0] IMPLICIT SEQUENCE {
         thisMech MechType,
         innerToken ANY DEFINED BY thisMech
             -- 32 byte initiator public key
    }
    END
```

On the first call to GSS_Init_sec_context(), the mechanism checks if one or more of the following are true:

The caller set anon_req_flag (GSS_C_ANON_FLAG)

The claimant_cred_handle identity is anonymous (see Section 4.1)

The claimant_cred_handle is the default credential and targ_name is anonymous

If none of the above are the case, the call MUST fail with GSS_S_UNAVAILABLE.

If proceeding, the initiator generates a fresh secret and public key pair per [RFC7748] Section 6.1 and returns GSS_S_CONTINUE_NEEDED, indicating that a subsequent context token from the acceptor is expected. The innerToken field of the output_token contains the initiator's 32 byte public key.

5.1.2. Acceptor context token

Upon receiving a context token from the initiator, the acceptor validates that the token is well formed and contains a public key of the requisite length. The acceptor generates a fresh secret and public key pair. The context session key is computed as specified in Section 6.

The acceptor constructs an output_token by concatenating its public key with the token emitted by calling GSS_GetMIC() with the default QOP and zero-length octet string. The output token is sent to the initiator without additional framing.

The acceptor then returns GSS_S_COMPLETE, setting src_name to the canonical anonymous name. The reply_det_state (GSS_C_REPLAY_FLAG), sequence_state (GSS_C_SEQUENCE_FLAG), conf_avail (GSS_C_CONF_FLAG),

integ_avail (GSS_C_INTEG_FLAG) and anon_state (GSS_C_ANON_FLAG) security context flags are set to TRUE. The context is ready to use.

5.1.3. Initiator context completion

Upon receiving the acceptor context token and verifying it is well formed, the initiator extracts the acceptor's public key (being the first 32 bytes of the input token) and computes the context session key per Section 6.

The initiator calls GSS_VerifyMIC() with the MIC extracted from the context token and the zero-length octet string. If successful, the initiator returns GSS_S_COMPLETE to the caller, to indicate the initiator is authenticated and the context is ready for use. No output token is emitted. Supported security context flags are as for the acceptor context. The flags returned to the caller are the intersection of supported and requested flags, combined with anon_state (GSS_C_ANON_FLAG) which is set unconditionally.

5.2. Per-Message Tokens

The per-message tokens definitions are imported from [RFC4121] Section 4.2. The base key used to derive specific keys for signing and sealing messages is defined in Section 6. The [RFC3961] encryption and checksum algorithms use the aes128-cts-hmac-sha256-128 encryption type defined in [RFC8009]. The AcceptorSubkey flag as defined in [RFC4121] Section 4.2.2 MUST be set.

5.3. Context Deletion Tokens

Context deletion tokens are empty in this mechanism. The behavior of GSS_Delete_sec_context() [RFC2743] is as specified in [RFC4121] Section 4.3.

6. Key derivation

The context session key is known as the base key, and is computed using a key derivation function from [SP800-108] Section 5.1 (using HMAC as the PRF):

```
base key = HMAC-SHA-256(K1, i | label | 0x00 | context | L)
```

where:

K1 the output of X25519(local secret key, peer public key) as specified in [RFC7748] Section 6.1

i the constant 0x00000001, representing the iteration count expressed in big-endian binary representation of 4 bytes

label the string "sanon-x25519" (without quotation marks)

context initiator public key | acceptor public key | channel binding application data (if present)

L the constant 0x00000080, being length in bits of the key to be outputted expressed in big-endian binary representation of 4 bytes

The inclusion of channel bindings in the key derivation function means that the acceptor cannot ignore initiator channel bindings; this differs from some other mechanisms.

The base key provides the acceptor-asserted subkey defined in [RFC4121] Section 2 and is used to generate keys for per-message tokens and the GSS-API PRF. Its encryption type is aes128-cts-hmac-sha256-128 per [RFC8009]. The [RFC3961] algorithm protocol parameters are as given in [RFC8009] Section 5.

7. Pseudo-Random Function

The [RFC4401] GSS-API pseudo-random function for this mechanism imports the definitions from [RFC8009], using the base key for both GSS_C_PRF_KEY_FULL and GSS_C_PRF_KEY_PARTIAL usages.

8. Security Considerations

This document defines a GSS-API security mechanism, and therefore deals in security and has security considerations text embedded throughout. This section only addresses security considerations associated with the SAnon mechanism described in this document. It does not address security considerations associated with the GSS-API itself.

This mechanism provides only for key agreement. It does not authenticate the identity of either party. It MUST NOT be selected if either party requires identification of its peer.

SAnon mechanism names are not unary. Implementations MUST ensure that GSS_Compare_name() always sets name_equal to FALSE when comparing mechanism names.

9. Acknowledgements

AuriStor, Inc funded the design of this protocol, along with an implementation for the Heimdal GSS-API library.

Jeffrey Altman, Greg Hudson, Simon Josefsson, and Nicolas Williams provided valuable feedback on this document.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[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>.

[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>.

[RFC4121] Zhu, L., Jaganathan, K., and S. Hartman, "The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2", RFC 4121, DOI 10.17487/RFC4121, July 2005, <https://www.rfc-editor.org/info/rfc4121>.

[RFC4401] Williams, N., "A Pseudo-Random Function (PRF) API Extension for the Generic Security Service Application Program Interface (GSS-API)", RFC 4401, DOI 10.17487/RFC4401, February 2006, <https://www.rfc-editor.org/info/rfc4401>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

[RFC8009] Jenkins, M., Peck, M., and K. Burgin, "AES Encryption with HMAC-SHA2 for Kerberos 5", RFC 8009, DOI 10.17487/RFC8009, October 2016, <https://www.rfc-editor.org/info/rfc8009>.

10.2. Informative References

[I-D.zhu-negoex]
Short, M., Zhu, L., Damour, K., and D. McPherson, "SPNEGO Extended Negotiation (NEGOEX) Security Mechanism", draft-zhu-negoex-04 (work in progress), January 2011.

[RFC4178] Zhu, L., Leach, P., Jaganathan, K., and W. Ingersoll, "The Simple and Protected Generic Security Service Application Program Interface (GSS-API) Negotiation Mechanism", RFC 4178, DOI 10.17487/RFC4178, October 2005, <https://www.rfc-editor.org/info/rfc4178>.

[RFC4757] Jaganathan, K., Zhu, L., and J. Brezak, "The RC4-HMAC Kerberos Encryption Types Used by Microsoft Windows", RFC 4757, DOI 10.17487/RFC4757, December 2006, <https://www.rfc-editor.org/info/rfc4757>.

[RFC5587] Williams, N., "Extended Generic Security Service Mechanism Inquiry APIs", RFC 5587, DOI 10.17487/RFC5587, July 2009, <https://www.rfc-editor.org/info/rfc5587>.

[RFC8062] Zhu, L., Leach, P., Hartman, S., and S. Emery, Ed., "Anonymity Support for Kerberos", RFC 8062, DOI 10.17487/RFC8062, February 2017, <https://www.rfc-editor.org/info/rfc8062>.

[SP800-108]
Chen, L., "Recommendation for Key Derivation Using Pseudorandom Functions (Revised)", October 2009.

Appendix A. Test Vectors

```
initiator secret key  69 df cc 04 2b 7a 33 f8 1a 43 fb f0 33 0a b5 3f
                      bc 20 e6 c1 4f f8 26 ce 6a 4d bc 8c 6e e4 2b a9

initiator public key  d2 1e 3e 58 60 b0 16 6c d1 cb 38 1a aa 89 62 93
                      07 13 ae e1 76 86 93 10 46 57 a7 a1 9c 1d 76 2e

initiator token       60 2c 06 0a 2b 06 01 04 01 a9 4a 1a 01 6e d2 1e
                      3e 58 60 b0 16 6c d1 cb 38 1a aa 89 62 93 07 13
                      ae e1 76 86 93 10 46 57 a7 a1 9c 1d 76 2e

acceptor secret key   3e 4f e6 5b ea 85 94 3b 5a a2 b7 83 f6 26 84 1a
                      10 39 d5 d3 6d af 85 aa a1 6f 12 97 57 99 6c ff

acceptor public key   a8 32 14 9d 58 33 13 ce 1c 55 7b 2b d1 8a e7 a5
                      59 8c a6 4b 02 20 83 5e 16 be 09 ca 2f 90 60 31
```

```
base key              af f1 8d b7 45 c6 27 cd a8 da d4 9b d7 e7 01 25

acceptor token        a8 32 14 9d 58 33 13 ce 1c 55 7b 2b d1 8a e7 a5
                      59 8c a6 4b 02 20 83 5e 16 be 09 ca 2f 90 60 31
                      04 04 05 ff ff ff ff ff 00 00 00 00 00 00 00 00
                      45 02 7b a8 15 1c 33 05 22 bb c4 36 84 d2 e1 8c
```

Appendix B. Mechanism Attributes

The [RFC5587] mechanism attributes for this mechanism are:

GSS_C_MA_MECH_CONCRETE

GSS_C_MA_ITOK_FRAMED

GSS_C_MA_AUTH_INIT_ANON

GSS_C_MA_AUTH_TARG_ANON

GSS_C_MA_INTEG_PROT

GSS_C_MA_CONF_PROT

GSS_C_MA_MIC

GSS_C_MA_WRAP

GSS_C_MA_REPLAY_DET

GSS_C_MA_OOS_DET

GSS_C_MA_CBINDINGS

GSS_C_MA_PFS

GSS_C_MA_CTX_TRANS

Appendix C. NegoEx

When SAnon is negotiated by [I-D.zhu-negoex], the authentication scheme identifier is DEE384FF-1086-4E86-BE78-B94170BFD376.

The initiator and acceptor keys for NegoEx checksum generation and verification are derived using the GSS-API PRF (see Section 7), with the input data "sanon-x25519-initiator-negoex-key" and "sanon-x25519-acceptor-negoex-key" respectively (without quotation marks).

The initiator metadata, if present, contains a set of GSS-API flags encoded as a 4 byte little endian integer. This is used to convey to the acceptor any Windows-specific GSS-API flags (see [RFC4757] Section 7.1). Other GSS-API flags MUST NOT be present in the metadata.

It is RECOMMENDED that GSS-API implementations supporting both SPNEGO [RFC4178] and NegoEx advertise SAnon under both to maximise interoperability.

Author's Address

Luke Howard
PADL Software Pty Ltd
PO Box 59
Central Park, VIC 3145
Australia

Email: lukeh@padl.com