Internet-Draft
Updates: RFC 1961
<draft-ietf-aft-socks-gssapi-revisions-01.txt>
Expires 24 December 1999

D. Miller
CyberSafe, Inc.

24 June 1999

# GSS-API Authentication Method for SOCKS Version 5

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC 2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Comments on this document should be sent to "aft@socks.nec.com", the IETF Authenticated Firewall Traversal WG discussion list. Distribution of this document is unlimited.

## 1. Abstract

The protocol specification for SOCKS Version 5 specifies a generalized framework for the use of arbitrary authentication protocols in the initial SOCKS connection setup. This document provides the specification for the SOCKS V5 GSS-API authentication protocol, and in particular, the use of the Simple and Protected GSS API Negotiation (SPNEGO) mechanism. Use of the SPNEGO pseudo-mechanism is intended to maximize the chance of agreement of a security mechanism, and hence maximize interoperability. A message protection subnegotiation protocol is also specified, allowing peers to agree on which message protection services GSS-API encapsulated messages will be protected with: integrity, or integrity and confidentiality. Other GSS-API security services, normally optional with GSS-API, are specified for use with SOCKS 5.

## 2. Introduction

Version 2 of the GSS-API, defined in [RFC 2078] and [draft-ietf-cat-gssv2-cbind-09.txt] provides an abstract interface which provides security services for use in distributed applications, but isolates

callers from specific security mechanisms and implementations.

The approach for use of GSS-API and SPNEGO in SOCKS V5 is to authenticate the client and server by successfully negotiating a common GSS-API mechanism and establishing a GSS-API security context - such that the GSS-API encapsulates any negotiation protocol for mechanism selection, and the agreement of security service options.

The SPNEGO mechanism defined in [RFC 2478] is a pseudo-security mechanism which enables GSS-API peers to determine in-band whether their credentials share common GSS-API security mechanism(s), and if so, to invoke normal security context establishment for a common security mechanism. The SPNEGO mechanism also allows the client to select a preferred mechanism, in a process referred to as optimistic negotiation, which eliminates extra negotiation round-trips in cases where the initiator and acceptor have the same preferred mechanism. This specification requires implementations to support the SPNEGO mechanism.

The GSS-API enables the context initiator to know what security services the target supports for the chosen mechanism. GSS-API mechanisms support message integrity. In addition to this, other optional services may be offered. These include message confidentiality, credential delegation, mutual authentication, replay detection, out-of-sequence detection, anonymous authentication, context transfer, and the use of incomplete contexts. As outlined later in this document, for successful context establishment, the following optional services are to be requested, with receipt of each service confirmed: credential delegation, mutual authentication, replay detection, and out-of-sequence detection. Those services are established as part of the security context establishment process proper. After context establishment, messages may either be protected for confidentiality or not, depending upon the input to the gss_wrap call. Message protection subnegotiation is the process whereby it is determined if message confidentiality is to be provided for the SOCKS session.

The GSS-API per-message protection calls gss_wrap and gss_unwrap are subsequently used to encapsulate any further TCP and UDP traffic between client and server.

## 3. Framing

The caller-opaque context establishment and per-message tokens produced by calls to the GSS-API are exchanged between client and server in the following format:

Where:

- "ver" is a single octet field representing the protocol version number, here 2 to represent the second version of the SOCKS/GSS-API

Protocol.

- "mtyp" is a single octet field representing the message type, which may contain the following values:
  0x01 - authentication message
  0x02 - message protection subnegotiation message
  0x03 - encapsulated user data

- "len" is a two octet field representing the length of the "token" field in octets.

- "token" is the opaque context establishment or per-message token emitted by the GSS-API, up to 2^16-1 octets in length.

## 4. SPNEGO Security Mechanism Negotiation

In order to effect SPNEGO, the client must specify SPNEGO as the mech_type parameter in its call to gss_init_sec_context. The OID for SPNEGO is iso.org.dod.internet.security.mechanism.snego (1.3.6.1.5.5.2).

The SPNEGO mechanism allows the client to optionally include a context establishment token for the preferred mechanism within the negotiation token, a process referred to as optimistic negotiation. If the client and server both support the same preferred mechanism, the negotiation and context establishment exchanges can occur simultaneously as described in [RFC 2479]. The SOCKS client side should use optimistic negotiation in order to maximize the efficiency of the connection to the SOCKS server.

Assuming optimistic negotiation is attempted, the client must specify use of the following security context options via the req_flags parameter in gss_init_sec_context: mutual authentication, credential delegation, replay detection, and out-of-sequence detection. The flags to include in the req_flags parameter that will request these service options are GSS_C_MUTUAL_FLAG, GSS_C_DELEG_FLAG, GSS_C_REPLAY_FLAG, and GSS_C_SEQUENCE_FLAG. GSS_C_SEQUENCE_FLAG should only be passed in for TCP-based clients, not for UDP-based clients. The client may optionally include GSS_C_CONF_FLAG or GSS_C_INTEG_FLAG into req_flags to request confidentiality and integrity services.

The negotiation tokens emitted by the GSS-API are exchanged between the client and server framed as described in section 3 with mtyp equal to 0x01.

Following the exchange of negotiation tokens and encapsulated mechanism tokens, and the successful establishment of a security context for the preferred mechanism as described in [RFC 2479], the client and server may proceed to the message protection subnegotation stage provided that confidentiality and integrity services are

available for the context (i.e. that GSS_C_CONF_FLAG and
GSS_C_INTEG_FLAG were returned in the ret_flags parameter of
gss_init_sec_context and gss_accept_sec_context)

If the client implementation does not have a preferred mechanism, or if the client and server do not support the same preferred mechanism, then the negotiation of a common security mechanism proceeds as defined in [RFC 2479].

If the client and server fail to agree on a common security mechanism, then the client must close the connection.

## 5. GSS-API Security Context Establishment

Clients which do not support a preferred mechanism, or clients whose preferred mechanisms were different than that of their peers, and which therefore were unable to perform optimistic negotiation (simultaneous negotiation and context establishment) must proceed to establish a security context with the negotiated mechanism. The client should proceed with security context establishment as defined in [RFC 2078]. In this case the negotiated mechanisms OID is used as the mech_type parameter in gss_init_sec_context. The same security context options used in section 4 should continue to be used.

## 6. GSS-API Protection-level Options

### 6.1 Message protection

Establishment of a GSS-API security context enables communicating peers to determine which per-message protection services are available to them. This is accomplished by inspecting the gss_init_sec_context and gss_accept_sec_context ret_flags parameters for the presence of the GSS_C_INTEG_FLAG and GSS_C_CONF_FLAG, which respectively indicate message integrity and confidentiality services are available. Note that while the indicated services are available, confidentiality, which is an optional service, is only applied if the appropriate flag is passed in to gss_wrap.

It is necessary to ensure that the message protection applied to the traffic is appropriate to the sensitivity of the data, and the severity of the threats. Not all message traffic needs to be protected for confidentiality, and avoiding this will make communications more efficient.

### 6.2 Message Protection Subnegotiation

For TCP and UDP clients and servers, different levels of protection are possible in the SOCKS V5 protocol, so an additional subnegotiation stage is needed to agree the message protection level. While this negotiation is part of neither SPNEGO nor GSS-API, its presence serves to increase interoperability between clients and servers that have

differing but flexible message protection policies.

After successful completion of this subnegotiation, TCP and UDP clients and servers use GSS-API encapsulation as defined in section 7.

After successful establishment of a GSS-API security context, the client's GSS-API implementation sends its required security context protection level to the server. The server then returns the security context protection level which it agrees to - which may or may not take the client's request into account. The security context protection level sent by client and server must be one of the following single-octet values:

0x01 - integrity-protected user data
0x02 - integrity-confidentiality-protected user data

The security context protection level is sent from client to server and vice versa framed as described in section 3 with an mtyp of 0x02.

### 6.3 Message Protection Subnegotiation Message Generation

The message protection subnegotiation message is a standard GSS-API message, and hence its processing results in a GSS-API token. The token is produced by encapsulating an octet containing the required protection level using gss_wrap with conf_req set to FALSE. The token is verified using gss_unwrap. If the server's choice of protection level is unacceptable to the client, then the client must close its connection to the server.

## 7. GSS-API Per-message Protection

For TCP and UDP clients and servers, the GSS-API functions for encapsulation and de-encapsulation shall be used by implementations, which are gss_wrap and gss_unwrap. (Note that the other GSS-API message protection functions, gss_get_mic and gss_verify_mic, do not encapsulate their messages, nor can they provide message confidentiality.)

When invoking gss_wrap and gss_unwrap, the default value of quality of protection shall be specified, and the use of conf_req_flag shall be as determined by the previous subnegotiation step. If protection level 1 is agreed then conf_req_flag MUST always be FALSE; if protection level 2 is agreed then conf_req_flag MUST always be TRUE.

All encapsulated messages are prefixed by the framing defined in section 3 with mtyp equal to 0x03.

## 8. GSS-API Security Context Termination

The GSS-API context termination message (emitted by gss_delete_sec_context) is not used by this protocol. When the connection is closed, each peer invokes gss_delete_sec_context passing GSS_C_NO_BUFFER as the value of the output_token argument. This suppresses production of the context termination message.

## 9. Open Issues

The viability of GSS-API authentication for the SOCKS protocol

needs to be reconciled with the current SOCKS mechanism negotiation scheme. That negotiation mechanism is unprotected.

The framing of socks protocol messages within the framing format described in this specification should be defined further.

Message protection subnegotiation might be unnecessary. Messages could be wrapped with the sender's choice of QOP and conf_req_flag values, as they are with other applications using GSS-API. Any such messages unacceptable to the recipient may be rejected, or the session may be terminated. The protection level evident by inspecting the context structures after security context establishment may effectively provide the subnegotiation discussed in this draft.

## 10. References

[RFC 1961] McMahon, P., "GSS-API Authentication Method for SOCKS Version 5", June 1996.

[SOCKS V5] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and L. Jones, "SOCKS Protocol V5", RFC 1928, April 1996.

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

[RFC 2478] Baize, E., Pinkas, D., "The Simple and Protected GSS-API Negotiation Mechanism," December, 1998

[draft-ietf-cat-gssv2-cbind-09.txt] Wray, J., "Generic Security Service API Version 2 : C-bindings", November 10, 1998.

## 11. Acknowledgment

The original document from which this document is derived is RFC 1961, written by P. McMahon, ICL. The first revisions document was written by David Margrave, CyberSafe Corporation. The revisions documents reflect input from the AFT WG.

## 12. Security Considerations

The protection features of SPNEGO require that all mechanisms proposed during the negotiation exchange support integrity services. If a single mechanism in the list does not support integrity, then the negotiation is subject to a downgrade attack.

The security services provided through the GSS-API are entirely dependent on the effectiveness of the underlying security mechanisms, and the correctness of the implementation of the underlying algorithms and protocols.

The user of a GSS-API service must ensure that the quality of protection provided by the mechanism implementation is consistent with their security policy.

In addition, where negotiation is supported under the GSS-API, constraints on acceptable mechanisms may be imposed to ensure suitability for application to authenticated firewall traversal.

## 13. Author's Address

David Miller
CyberSafe Corporation
1605 NW Sammamish Road, Suite 310
Issaquah, Washington 98027 USA

Email: david.miller@cybersafe.com
Phone: (425) 391-6000
Fax: (425) 391-0508

Document expires 24 December 1999