<draft-ietf-aft-gssapi-00.txt> P V McMahon, ICL

GSS-API Authentication Method for SOCKS Version 5

Status of this Memo

Comments on this document are welcome and should be sent to aft@unify.com, the mailing list of the Authenticated Firewall Traversal Working Group of the IETF.

Contents List

1. Purpose

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 defines an GSS-API authentication method encapsulation that provides integrity, authentication and optional confidentiality.

2. Introduction

GSS-API provides an abstract interface which provides security services for use in distributed applications, but isolates callers from specific security mechanisms and implementations.

GSS-API peers achieve interoperability by establishing a common security mechanism for security context establishment - either through administrative action, or through negotiation. GSS-API is specified in [RFC 1508], and [RFC 1509].

The approach for use of GSS-API in SOCKS V5 is to authenticate the client and server by successfully 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 GSS-API gss_init_sec_context() interface enables the context initiator to know what security services the target supports for the chosen mechanism.

The GSS-API per-message protection calls are used to encapsulate any further TCP traffic between client and server, and, for integrity protection of UDP datagrams.

3. GSS-API Call Specification for SOCKS V5

3.1 Preparation

Prior to use of GSS-API primitives, the client and server should be locally authenticated, and have established GSS-API credentials.

The client should call gss_import_name to obtain an internal representation of the server name. For maximal portability the default name_type GSS_C_NULL_OID should be used to specify the default name space, and the input name_string should treated by the client as an opaque name-space specific input. For example, when using Kerberos V5 naming, the imported name is of the form "SERVICE:socks@socks_server_hostname" where "socks_server_hostname" is the fully qualified host name of the server with all letters in lower case.

3.2 Client Context Establishment

The client should then call gss_init_sec_context, typically passing GSS_C_NO_CREDENTIAL into cred_han to specify the default credential (for initiator usage), GSS_C_NULL_OID into mech_type to specify the default mechanism, GSS_C_NO_CONTEXT into context_handle to specify a NULL context (initially), and the previously imported server name into targ_name.

The client must also specify its requirements for replay protection, delegation, and sequence protection via the gss_init_sec_context req_flags parameter. It is required by this specification that the client always requests these service options (i.e. passes GSS_C_MUTUAL_FLAG | GSS_C_REPLAY_FLAG | GSS_C_DELEG_FLAG | GSS_C_MUTUAL_FLAG into req_flags). However, GSS_C_SEQUENCE_FLAG should only be passed in for TCP-based clients, not for UDP-based clients.

3.3 Client Context Establishment Major Status codes

The gss_init_sec_context returned status code can take two different success values:

- If gss_init_sec_context returns GSS_S_CONTINUE_NEEDED, then the client should expect the server to issue a token in the subsequent subnegotiation response. The client must pass the token to another call to gss_init_sec_context, and repeat this procedure until continue operations are complete.

- If gss_init_sec_context returns GSS_S_COMPLETE, then the client should respond to the server with any resulting output_token. If there is no output_token, the client should proceed to sending the protected request details.

3.4 Client initial token

The client's GSS-API implementation then typically responds with the resulting output_token which the client sends in a message to the server.

```
+------+------+------+......................+
+ ver  | mtyp | len  |       token          |
+------+------+------+......................+
+ 0x01 | 0x01 | 0x02 | up to 2^16 - 1 octets |
+------+------+------+......................+
```

If, however, the client's GSS-API implementation failed during gss_init_sec_context, the the client must close its connection to the server.

3.5 Server Context Establishment

For the case where a client successfully sends a token emitted by gss_init_sec_context() to the server, the server must pass the client-supplied token to gss_accept_sec_context as input_token.

For portability, verifier_cred_handle is set to GSS_C_NO_CREDENTIAL (for acceptor usage), context_handle initially set to GSS_C_NO_CONTEXT.

If gss_accept_sec_context returns GSS_CONTINUE_NEEDED, the server should return the generated output_token to the client, and subsequently pass the resulting client supplied token to another call to gss_accept_sec_context.

If gss_accept_sec_context returns GSS_S_COMPLETE, then if an output_token is returned, the server should return it to the client. If no token is returned, a zero length token should be sent by the server to signal to the client that it is ready to receive the client's request.

3.6 Server Reply

In all continue/confirmation cases, the server uses the same message type as for the client -> server interaction.

```
+------+------+------+.......................+
+ ver  | mtyp | len  |       token           |
+------+------+------+.......................+
+ 0x01 | 0x01 | 0x02 | up to 2^16 - 1 octets |
+------+------+------+.......................+
```

3.7 Security Context Failure

If the server refuses the client's connection for any reason (GSS-API authentication failure or otherwise), it will return:

```
+------+------+
+ ver  | mtyp |
+------+------+
+ 0x01 | 0xff |
+------+------+
```

3.8 UDP Protection

When using GSS-API, the authentication key material identified in [SOCKS V5] for computation of the value for the XCOOKIE digest within the UDP MAC field is encapsulated by the authentication mechanism.

Therefore, for UDP-based clients, the XCOOKIE digest value for UDP is derived by invoking gss_get_mic() for the COOKIE from the UDP ASSOCIATE request.

4. References

[RFC 1508] Generic Security Service API, J Linn,
September 1993

[RFC 1509] Generic Security Service API : C-bindings, J Wray,
September 1993

[SOCKS V5] SOCKS Protocol V5, draft-ietf-aft-socks-proto-v5-01.txt
M Leech, March 1995

5. Acknowledgment

This document builds from a previous draft produced by Marcus Leech (BNR) - whose comments are gratefully acknowleged.

6. Security Considerations

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.

7. Author's Address

P V McMahon
post: ICL Enterprises, Kings House, 33 Kings Road, Reading, RG1 3PX, UK
email: p.v.mcmahon@rea0803.wins.icl.co.uk
phone: +44 734 634882
fax: +44 734 855106