Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

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/lid-abstracts.txt.

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

This Internet-Draft will expire on January 14, 2009.
Abstract

This document specifies an extension to the HTTP Negotiate authentication mechanism defined in RFC4559 which supports mutual authentication, fast session-based reauth and channel bindings.

Table of Contents

1. Terminology ................................................. 3
2. Introduction and motivation .............................. 4
3. HTTP GSS Authentication Mechanism ...................... 5
   3.1. GSS Token Header Syntax ............................. 5
   3.2. Naming and Transport ................................ 5
   3.3. Protocol Flow ......................................... 6
       3.3.1. Initiating authentication ....................... 6
       3.3.2. The authentication phase ....................... 6
       3.3.3. The authorization phase ....................... 8
       3.3.4. Fast Renegotiation ................................ 8
4. Examples ................................................... 10
5. Implementation Notes ........................................ 11
6. Security Considerations .................................... 12
7. Notes & TODO ............................................... 13
8. IANA Considerations ........................................ 14
9. Changes ..................................................... 15
   9.1. 00 to 01 ............................................... 15
   9.2. 02 to 03 ............................................... 15
10. References .................................................. 16
    10.1. Normative References ................................. 16
    10.2. Informative References ............................... 16
Author’s Address ............................................... 17
Intellectual Property and Copyright Statements .............. 18
1. Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT" and "MAY" that appear in this document are to be interpreted as described in [RFC2119]
2. Introduction and motivation

[ RFC4559 ] describes an authentication mechanism based on SPNEGO for HTTP. This mechanism suffers from a couple of drawbacks, notably:

- Only supports single-round-trip GSS-API mechanisms
- Lack of channel bindings to the underlying HTTPS connection which makes it unsuitable for deployment in situations where proxies exist.
- Lack of session-based re-authentication (compare with TLS).

This document is intended to solve these problems by introducing a new authentication mechanism called 'GSS'. This mechanism is a proper extension of Negotiate but since Negotiate is already widely deployed this mechanism was given a separate name.
3. HTTP GSS Authentication Mechanism

The GSS mechanism is an authentication mechanism for [RFC2616] based on a multi-roundrip handshake using base64-encoded GSS-API [RFC2743] tokens encoded in the WWW-Authorenticate Response Header and the Authorization Request Header. An important difference from [RFC4559] is that multiple roundtrips are supported which means that the server can be authenticated to the client (aka mutual authentication).

3.1. GSS Token Header Syntax

Both the Authorization and the WWW-Authenticate headers use the same syntax throughout the handshake (cf below for details on the protocol flow) specified by this Augmented BNF following [RFC2617] and [RFC2616]:

```
challenge = auth-scheme 1*SP 1#auth-param
auth-scheme = "GSS"
auth-param = auth-param-type "=" auth-param-value
auth-param-value = ( token | quoted-string )
auth-param-type = ( "auth-data" | "context-identifier" )
auth-data-value = 1*(UPALPHA|DIGIT) ;base64-encoded
context-identifier-value = 1*(UPALPHA|DIGIT) ;base64-encoded
```

The auth-param types defined by this specification (auth-data and context-identifier) both have auth-param-value which contain base64 encoded data. Note that both the auth-data and context-identifier auth-param may be absent. The semantics of these parameters will be explained below. Each auth-param-type MUST NOT occur more than once in a single challenge.

[NOTE: It may be interesting to allow existing implementations of NEGOTIATE to treat the fast reauth feature as an optional extension. In this case it will be necessary to change to way the authentication token is represented.]

3.2. Naming and Transport

The GSS name of the server is "HTTP@<hostname>[:port]" where the :port part is absent if the port == 80 or if the port == 443.

This mechanism SHOULD be used together with an HTTP transport providing session protection and encryption such as [RFC2817] or [RFC2818]. Session protection is a requirement for fast reauthentication described below.

Like [RFC4559] the mechanism described in this specification is based on mapping the GSS-API protocol to HTTP requests and responses where
the GSS-API tokens are sent in the Authorization and WWW-
Authentication headers. Unlike [RFC4559] the entire handshake need
not take place using a single TCP connection or a single HTTP/1.1
session. Instead opaque identifiers in the GSS challenge option
field are optionally used together with channel bindings to provide a
way to share a security context over several HTTP connections. This
mechanism also serves as a way to let the client do a fast
reauthentication to the server.

3.3. Protocol Flow

3.3.1. Initiating authentication

Normally the server initiates an authentication handshake when the
client attempts to access a protected resource. The exception is
when the client knows that it is accessing a protected resource and
that the server supports the GSS mechanism, for instance when fast
re-authentication is attempted by the client (cf below). In both
cases the GSS-API negotiation is initiated by the client - i.e if the
server initiates the authentication it is only to inform the client
that authentication is required. The client SHOULD request mutual
authentication from the GSS-API layer.

Note that the first request by the client to a protected resource
will also serve to let the client and server establish channel
bindings using the 'tls-server-end-point' CB type which means that
this first request is not in general "wasted" even in the case when
the client has no prior knowledge about the server or is attempting
fast re-authentication.

If the client tries to access a protected resource the server may
return a code 401 response with an WWW-Authenticate header containing
a list of authentication challenges allowing the client to choose
among different authentication mechanisms supported by the server.
If the server supports the "GSS" mechanism the server returns a
challenge with only the auth-scheme part ("GSS") and no parameters
along with any other challenges for mechanisms supported by the
server. This first request also allows the client and server to
establish channel-bindings.

3.3.2. The authentication phase

In each case below when GSS-API tokens resulting from calls into the
GSS-API layer are sent from the server to the client or vice-versa,
the token is encoded using base64 and sent as the "auth-data"
parameter value of the Authorization and WWW-Authenticate headers
respectively.
A client initiates the authentication phase by sending the token resulting from the first call to gss_initiate_security_context to the server.

Upon receipt of token (i.e. a request with an accompanying Authenticate header with non-empty "auth-data" parameter value), the server MUST return the token resulting from a call to gss_accept_security_context in a code 401 response, unless the call to gss_accept_security_context fails in which case a code 403 response is returned.

If the underlying transport provides session protection (e.g., HTTPS) and if channel-bindings are in place (cf below) then the server MAY include a unique identifier of the security context being negotiated (or having been negotiated in the case of the last transaction) in the "context-identifier" parameter value. The server MUST uniquely associate this identifier with the client and the security context.

Upon receipt of a code 401 response from the server when the WWW-Authenticate header contains a non-empty "auth-data" parameter value, the client MUST return the token resulting from a call to gss_initiate_security_context to the server in a new request to the same resource. If the call fails the client MUST close the connection. If a "context-identifier" parameter value is present in the response from the server the client MUST include this in the ensuing request as the "context-identifier" parameter value. If the "context-identifier" parameter value is not present in the response from the server the client MUST use the same HTTP/1.1 connection for the entire handshake. If the client breaks the HTTP/1.1 connection the server MUST invalidate the security context unless a context identifier was sent to the client and returned to the server.

A client may close the connection both as the result of using the context-identifier to spread the authentication over several underlying connections or as the result of a failed call to gss_initiate_security_context. This might at first seem like a problem but the GSS-API layer combined with proper handling of the context identifier will ensure that handling of these cases are disambiguated at the server.

The client and server continues the handshake until either an error occurs (in which case a 403 is returned to the client or the client closes the connection depending on where the error happens) or the GSS-API layer has successfully completed the negotiation in which case the server sends a normal response to the client. If the last call to gss_accept_sec_context on the server resulted in a non-empty token the server MUST include this in a WWW-Authenticate header in the response to the client regardless of the return code which is
beeing sent to the client. If the underlying transport provides session protection (e.g., HTTPS) and if channel-bindings are in place (cf. below) then the server MAY include a "context-identifier" parameter value uniquely identifying the established security context. The server MAY decide to limit the validity of the established context and MAY choose not to consider references to the context after a certain amount of time (cf. below).

If the client receives a normal response with a non-empty "auth-data" parameter value the client MUST call gss_initiate_sec_context with this token as input to complete the authentication handshake. If the final response contains a "context-identifier" parameter value the client may cache it and use it to provide fast re-authentication by including it in an Authorization header with "GSS" mechanism and empty "auth-data" parameter value.

3.3.3. The authorization phase

Authorization failures can occur even if the client is successfully authenticated to the server. In this case the server will send a 403 response to the client even though the GSS-API handshake has succeeded. It is important to let the client and server finish the authentication handshake even if the client is not authorized to access the resource. Therefore the client MUST call gss_initiate_sec_context with any GSS-API token returned to the client, even if the token was sent along with a 403 response.

During authorization the server MAY use the GSS-API name associated with the established security context for authorization decisions and SHOULD provide a string representation of the GSS-API name as the REMOTE_USER meta-variable and "GSS" as the AUTH_TYPE meta-variable if the Common Gateway Interface is provided by the server.

3.3.4. Fast Renegotiation

Upon receipt of a request containing an Authorization header with the "GSS" mechanism, an empty auth-data and the context-identifier parameter value, the server MUST verify that the identifier references a valid security context. If the security context is missing or invalid the server MUST return a 401 response prompting the client to re-negotiate the security context. If the identifier references a valid security context the server MUST process the request as if the client had just completed the full authentication handshake.

When this process is completed the client is authenticated to the server and possibly (depending on the way the GSS-API layer was called and which GSS-mechanism was used) the server is authenticated.
to the client.

The use of fast reegotiation is optional and clients and servers MUST NOT assume that this feature is supported.
4. Examples

TODO
5. Implementation Notes

The context-identifier could be produced by exporting the security context through gss_export_sec_context which requires that the GSS-API implementation supports exporting unfinished contexts.
6. Security Considerations

Should channel-bindings be absent, the protocol is subject to a MITM attack unless the authentication is between a client and a server with no proxies in between and each request is sent over the same HTTP/1.1 connection.

If fast re-authentication is used together with GSS-API credentials delegation the server will need to associate forwarded credentials with the negotiated security context. This presents a challenge for server implementors since it must be guaranteed that security states and their associated credentials must be separated from each other.
7. Notes & TODO

Write a specification of how the client and server uses 
draft-johansson-http-tls-cb.

Rewrite the CGI language as non-normative text.
8. IANA Considerations

Registration of the GSS HTTP authentication mechanism?
9. Changes

9.1. 00 to 01

Changed from ABNF to Augmented BNF to align with [RFC2616].

9.2. 02 to 03

Added reference to rfc 5056.

Reference to tls-server-end-point channel binding mechanism.
10. References

10.1. Normative References


10.2. Informative References


Author’s Address

Leif Johansson  
Stockholm university

Email: leifj@it.su.se  
URI:  http://www.su.se/
Full Copyright Statement

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.