Network Working Group	Johansson	
Internet-Draft	SU	
Intended status: Standards Track	March 08, 2009	
Expires: September 9, 2009		

GSSAPI authentication for HTTP
draft-johansson-http-gss-05

Status of this Memo

Copyright Notice

Abstract

This document specifies a template extension to the HTTP Negotiate authentication mechanism defined in RFC4559 which supports mutual authentication, fast session-based re-authentication and channel bindings. An IANA registry for such GSS-API HTTP authentication mechanisms is defined.

Table of Contents

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] (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.)

2. Introduction and motivation

[RFC4559] (Jaganathan, K., Zhu, L., and J. Brezak, “SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows,” June 2006.) 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 due to lack of support for proxies.
>
> Lack of channel bindings to the underlying HTTPS connection which makes in unsuitable for deployment in situations where proxies exists.
>
> 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

TOC

The GSS mechanism is an authentication mechanism for [RFC2616] (Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, “Hypertext Transfer Protocol -- HTTP/1.1,” June 1999.) based on a multi-roundrip handshake using base64-encoded GSS-API [RFC2743] (Linn, J., “Generic Security Service Application Program Interface Version 2, Update 1,” January 2000.) tokens encoded in the WWW-Authenticate Response Header and the Authorization Request Header. An important difference from [RFC4559] (Jaganathan, K., Zhu, L., and J. Brezak, “SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows,” June 2006.) is that multiple round trips are supported which means that the server can be authenticated to the client (aka mutual authentication). This document specifies a template authentication mechanism with an associated IANA registry which provides input parameters to the HTTP authentication mechanism describe below.

3.1. GSS Token Header Syntax

TOC

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] (Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, “HTTP Authentication: Basic and Digest Access Authentication,” June 1999.) and [RFC2616] (Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, “Hypertext Transfer Protocol -- HTTP/1.1,” June 1999.):

```
challenge                 = auth-scheme-name 1*SP 1#auth-param
auth-scheme-name          = token
auth-param                = ( auth-data-value |
                              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.
The auth-scheme-name token is the name of the mechanism and is supplied in the IANA registry template described below.
For reasons of backwards compatibility with [RFC4559] (Jaganathan, K., Zhu, L., and J. Brezak, “SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows,” June 2006.) two forms of the auth-param are allowed - one version based on attribute-parameter pairs and one where only GSS-API data is sent.

3.2. Naming and Transport

TOC

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] (Khare, R. and S. Lawrence, “Upgrading to TLS Within HTTP/1.1,” May 2000.) or [RFC2818] (Rescorla, E., “HTTP Over TLS,” May 2000.) . Session protection is a requirement for fast re-authentication described below.
Like [RFC4559] (Jaganathan, K., Zhu, L., and J. Brezak, “SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows,” June 2006.) 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] (Jaganathan, K., Zhu, L., and J. Brezak, “SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows,” June 2006.) 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 re-authentication to the server.

3.3. Protocol Flow

TOC

3.3.1. Intiating authentication

TOC

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 in the sense of [RFC5056] (Williams, N., “On the Use of Channel Bindings to Secure Channels,” November 2007.) 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 mechanism specified by the auth-scheme-name the server returns a challenge with only the auth-scheme-name part 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

TOC

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_init_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 (eg HTTPS) and
if channel-bindings are in place (cf below) then the server MAY include
a unique identifier of the security context beeing 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 entier 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 (eg
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 an 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 a Authorization header with auth-scheme-name and empty "auth-data" parameter value.

3.3.3. The authorization phase

TOC

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 the auth-scheme-name as the AUTH_TYPE meta-variable if the Common Gateway Interface (CGI) is provided by the server.

3.3.4. Fast Renegotiation

TOC

Upon receipt of a request containing an Authorization header with the auth-scheme-name, 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 regegotiation is optional and clients and servers MUST NOT assume that this feature is supported.

---

4. Examples

TOC

TODO

5. Implementation Notes

TOC

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

TOC

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 guarateed that security states and their associated credentials must be separated from each other.

7. Notes & TODO

TOC

Examples

8. IANA Considerations

TOC

IANA will create a new registry for HTTP authentication mechanisms based on this document. The purpose of the registry is to bind the HTTP authentication mechanism name (auth-scheme-name in the syntax above) to the GSS-API mechanism OID. Such HTTP authentication mechanisms will be called GSS-API HTTP authentication mechanisms.
Names for GSS-API HTTP authentication mechanisms must follow the token syntax of section 2.2 of [RFC2616] (Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, “Hypertext Transfer Protocol -- HTTP/1.1,” June 1999.).
The procedure detailed in the section below is to be used for registration of a value naming a specific GSS-API HTTP authentication mechanism.

TOC

8.1. Registration Procedure

Registration of a new GSS-API HTTP authentication mechanism requires expert review as defined in BCP 26 [RFC2434] (Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs," October 1998.). Registration of a GSS-API HTTP authentication mechanism is requested by filling in the following template:

> Subject: Registration of GSS-API HTTP authentication mechanism X
>
> GSS-API HTTP authentication mechanism name:
>
> GSS-API mechanism OID:
>
> Description or Published Specification:
>
> State management: (one of INTERNAL or EXTERNAL)
>
> Intended usage: (one of COMMON, LIMITED USE, OBSOLETE)
>
> Person and email address to contact for further information:
>
> Change manager name and email address:
>
> Expert reviewer name and contact information: (leave blank)
>
> Note: (Any other information deemed relevant)

and sending it via electronic mail to <gss-http@ietf.org> (a public mailing list) and carbon copying (cc:) IANA at <iana@iana.org>. After allowing new fewer than 2 weeks for community input on the mailing list to be determined, an expert will determine the appropriateness of the registration request and either approve or disapprove the request with notice to the requester, the mailing list and IANA.
If the registration was approved the expert adds her name to the submitted registration.
The expert is responsible for making sure that GSS-API authentication scheme names are unique among all HTTP authentication mechanism names and represent an appropriate name for the underlying GSS-API mechanism.
Authors are encouraged to pursue community review by posting the technical specification as an Internet-Draft and soliciting comment by posting to appropriate IETF mailing lists.

8.2. Change Control

TOC

Once a GSS-API HTTP authentication mechanism has been published by IANA, the author may request a change to its definition. The change

request follows the same procedure as the registration request. The change manager is part of the registration template and controls who may request changes to the registration. Passing control of a registration is also accomplished by submitting a change request.
The IESG may also reassign control and responsibility for GSS-API HTTP authentication mechanism registrations. This is expected to happen when the author of a registration has died, has moved out of contact, or is otherwise unable to make changes to the registered mechanism(s)s. Furthermore the IESG is the owner of all GSS-API HTTP authentication mechanisms that correspond to specifications on the IETF standards track.

9. Changes

TOC

9.1. 00 to 01

TOC

Changed from ABNF to Augmented BNF to align with [RFC2616] (Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, “Hypertext Transfer Protocol -- HTTP/1.1,” June 1999.).

9.2. 02 to 03

TOC

Added reference to rfc 5056.
Reference to tls-server-end-point channel binding mechanism.

9.3. 03 to 04

TOC

Generalized to IANA-controlled registry of authentication mechanisms. Wrote IANA considerations section. Generalized the ABNF to cover old Negotiate case which can now be turned into an IANA registration covered by this specification.

10. References

TOC

10.1. Normative References

TOC

[RFC2119]	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[RFC2434]	Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs," BCP 26, RFC 2434, October 1998 (TXT, HTML, XML).
[RFC2616]	Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1," RFC 2616, June 1999 (TXT, PS, PDF, HTML, XML).
[RFC2617]	Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication," RFC 2617, June 1999 (TXT, HTML, XML).
[RFC2743]	Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1," RFC 2743, January 2000 (TXT).
[RFC2817]	Khare, R. and S. Lawrence, "Upgrading to TLS Within HTTP/1.1," RFC 2817, May 2000 (TXT).
[RFC2818]	Rescorla, E., "HTTP Over TLS," RFC 2818, May 2000 (TXT).
[RFC5056]	Williams, N., "On the Use of Channel Bindings to Secure Channels," RFC 5056, November 2007 (TXT).

10.2. Informative References

TOC

[RFC4559]	Jaganathan, K., Zhu, L., and J. Brezak, "SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows," RFC 4559, June 2006 (TXT).

Author's Address

TOC

	Leif Johansson
	Stockholm university
Email:	leifj@it.su.se
URI:	http://www.su.se/