Kerberos working group
Internet Draft
Document: draft-brezak-spnego-http-04.txt
Category: Informational

J.Brezak
Microsoft

October 2002

HTTP Authentication: SPNEGO Access Authentication As implemented in Microsoft Windows 2000

Status of this Memo

1. Abstract

This document describes how the Microsoft Internet Explorer (MSIE) and Internet Information Services (IIS) incorporated in Microsoft Windows 2000 use Kerberos for security enhancements of web transactions. The HTTP auth-scheme of "negotiate" is defined here; when the negotiation results in the selection of Kerberos, the security services of authentication and optionally impersonation are performed.

This document explains how HTTP authentication utilizes the SPNEGO [7] GSSAPI mechanism. Details of SPNEGO implementation are not provided in this document.

2. Conventions used in this document

In examples, "C:" and "S:" indicate lines sent by the client and server respectively.

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 RFC-2119 [3].

3. Access Authentication

3.1 Reliance on the HTTP/1.1 Specification

This specification is a companion to the HTTP/1.1 specification [4] and builds on the authentication mechanisms defined in [5]. It uses the augmented BNF section 2.1 of that document, and relies on both the non-terminals defined in that document and other aspects of the HTTP/1.1 specification.

4. HTTP Negotiate Authentication Scheme

Use of Kerberos is wrapped in an HTTP auth-scheme of "Negotiate". The auth-params exchanged use data formats defined for use with the GSS-API [6]. In particular, they follow the formats set for the SPNEGO [7] and Kerberos [8] mechanisms for GSSAPI. The "Negotiate" auth-scheme calls for the use of SPNEGO GSSAPI tokens which the specific mechanism type specifies.

The current implementation of this protocol is limited to the use of SPNEGO with the Kerberos and Microsoft NTLM protocols.

4.1 The WWW-Authenticate Response Header

If the server receives a request for an access-protected object, and an acceptable Authorization header has not been sent, the server responds with a "401 Unauthorized" status code, and a "WWW-Authenticate:" header as per the framework described in [4]. The initial WWW-Authenticate header will not carry any gssapi-data.

The negotiate scheme will operate as follows:

```
challenge       = "Negotiate" auth-data
auth-data       = 1#( [gssapi-data] )
```

The meanings of the values of the directives used above are as follows:

gssapi-data
: If the gss_accept_security_context return a token for the client, this directive contains the base64 encoding of an InitialContextToken as defined in [6]. This is not present in the initial response from the server.

A status code 200 status response can also carry a "WWW-Authenticate" response header containing the final leg of an authentication. In this case, the gssapi-data will be present. Before using the contents of the response, the gssapi-data should be processed by gss_init_security_context to determine the state of the security context. If this function indicates success, the response can be used by the application. Otherwise an appropriate action based on the authentication status should be.

For example the authentication could have failed on the final leg if mutual authentication was requested and the server was not able to prove its identity. In this case, the returned results are suspect. It is not always possible to mutually authenticate the server before the HTTP operation. POST methods are in this category.

When the Kerberos Version 5 GSSAPI mechanism [RFC-1964] is being used, the HTTP server will be using a principal name of the form of "HTTP/<hostname>".

4.2 The Authorization Request Header

Upon receipt of the response containing a "WWW-Authenticate" header from the server, the client is expected to retry the HTTP request, passing a HTTP "Authorization" header line. This is defined according to the framework described in [4] utilized as follows:

```
credentials             = "Negotiate" auth-data2
auth-data2              = 1#( gssapi-data )
```

gssapi-data
: This directive contains is the base64 encoding of an InitialContextToken as defined in [6].

Any returned code other than a success 2xx code represents an authentication error. If a 401 containing a "WWW-Authenticate" header with "Negotiate" and gssapi-data is returned from the server, it is a continuation of the authentication request.

A client may initiate a connection to the server with an "Authorization" header containing the initial token for the server. This form will bypass the initial 401 error from the server when the client knows that the server will accept the Negotiate HTTP authentication type.

5. Negotiate Operation Example

The client requests an access-protected document from server via a GET method request. The URI of the document is "http://www.nowhere.org/dir/index.html".

```
C: GET dir/index.html
```

The first time the client requests the document, no Authorization header is sent, so the server responds with:

```
S: HTTP/1.1 401 Unauthorized
S: WWW-Authenticate: Negotiate
```

The client will obtain the user credentials using the SPNEGO GSSAPI mechanism type to identify generate a GSSAPI message to be sent to the server with a new request, including the following Authorization

header:


```
C: GET dir/index.html
C: Authorization: Negotiate a87421000492aa874209af8bc028
```

The server will decode the gssapi-data and pass this to the SPNEGO GSSAPI mechanism in the gss_accept_security_context function. If the context is not complete, the server will respond with a 401 status code with a WWW-Authenticate header containing the gssapi-data.

```
S: HTTP/1.1 401 Unauthorized
S: WWW-Authenticate: Negotiate 749efa7b23409c20b92356
```

The client will decode the gssapi-data and pass this into gss_init_security_context and return the new gssapi-data to the server.

```
C: GET dir/index.html
C: Authorization: Negotiate 89a8742aa8729a8b028
```

This cycle can continue until the security context is complete.

When the return value from the gss_accept_security_context function indicates that the security context is complete, it may supply final authentication data to be returned to the client. If the server has more gssapi data to send to the client to complete the context it is to be carried in WWW-Authenticate header with the final response containing the HTTP body.

```
S: HTTP/1.1 200 Success
S: WWW-Authenticate: Negotiate ade0234568a4209af8bc0280289eca
```

The client will decode the gssapi-data and supply it to gss_init_security_context using the context for this server. If the status is successful from the final gss_init_security_context, the response can be used by the application.

7. Security Considerations

The SPNEGO HTTP authentication facility is only used to provide authentication of a user to server. It provides no facilities for protecting the HTTP headers or data including the Authorization and WWW-Authenticate headers that are used to implement this mechanism.

This mechanism is not used for HTTP authentication to HTTP proxies.

If an HTTP proxy is used between the client and server, it must take care to not share authenticated connections between different authenticated clients to the same server. If this is not honored, then the server can easily lose track of security context associations. A proxy that correctly honors client to server authentication integrity will supply the "Proxy-support: Session-

Based-Authentication" HTTP header to the client in HTTP responses from the proxy. The client MUST NOT utilize the SPNEGO HTTP authentication mechanism through a proxy unless the proxy supplies this header with the "401 Unauthorized" response from the server.

When using the SPNEGO HTTP authentication facility with client supplied data such as PUT and POST, the authentication should be complete between the client and server before sending the user data. The return status from the gss_init_security_context will indicate with the security context is complete. At this point the data can be sent to the server.

8. References

1 Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

3 Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997

4 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.

5 Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., Stewart, L., "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

6 Linn, J., "Generic Security Service Application Program Interface, Version 2", RFC 2078, January 1997.

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

8 Linn, J., "The Kerberos Version 5 GSS-API Mechanism", RFC 1964, June 1996.

10. Author's Addresses

John Brezak
Microsoft
One Microsoft Way
Redmond, Washington
Email: jbrezak@microsoft.com

Full Copyright Statement