              SPNEGO-based Kerberos and NTLM HTTP Authentication
                            in Microsoft Windows

Status of This Memo

   This memo provides information for the Internet community. It does
   not specify an Internet standard of any kind. Distribution of this
   memo is unlimited.

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 Hypertext Transport Protocol (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 (the IIS server assumes the windows
   identity of the principal that has been authenticated) are performed.
   This document explains how HTTP authentication utilizes the Simple
   and Protected GSS-API Negotiation mechanism. Details of Simple And
   Protected Negotiate (SPNEGO) implementation are not provided in this
   document.

Table of Contents

1. Introduction

   Microsoft has provided support for Kerberos authentication in
   Microsoft Internet Explorer (MSIE) and Internet Information Services
   (IIS), in addition to other mechanisms. This provides the benefits
   of the Kerberos v5 protocol for Web applications.

   Support for Kerberos authentication is based on other previously
   defined mechanisms, such as SPNEGO Simple And Protected Negotiate
   (SPNEGO) [RFC4178] and the Generic Security Services Application
   Program Interface(GSSAPI).

2. Conventions Used in This Document

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to
   be interpreted as described in [RFC2119].

3. Access Authentication

3.1. Reliance on the HTTP/1.1 Specification

   This specification is a companion to the HTTP/1.1 specification
   [RFC2616], and it builds on the authentication mechanisms defined in
   [RFC2617]. It uses the augmented BNF section of that document (2.1),
   and it 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 [RFC2743]. In particular, they follow the formats set for
   the SPNEGO [RFC4178] and Kerberos [RFC4121] mechanisms for GSSAPI.
   The "Negotiate" auth-scheme calls for the use of SPNEGO GSSAPI tokens
   that the specific mechanism type specifies.

   The current implementation of this protocol is limited to the use of
   SPNEGO with the Kerberos and Microsoft(NT Lan Manager) NTLM
   protocols.

4.1. The WWW-Authenticate Response Header

   If the server receives a request for an access-protected object, and
   if 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 [RFC2616].
   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 returns a token for the client,
this directive contains the base64 encoding of an
initialContextToken, as defined in [RFC2743]. 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 taken.

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 [RFC4121] 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 [RFC2616] and is utilized as
follows:

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

gssapi-data

This directive contains the base64 encoding of an
InitialContextToken, as defined in [RFC2743].

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, 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 a 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.

6. Security Considerations

The SPNEGO HTTP authentication facility is only used to provide
authentication of a user to a 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.

Alternate mechanisms such as TLS can be used to provide
confidentiality. Hashes of the TLS certificates can be used as
channel bindings to secure the channel. In this case clients would
need to enforce that the channel binding information is valid. Note
that Kerb-TLS [RFC2712] could be used to provide both authentication
and confidentiality, but this requires a change to the TLS provider.

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 that the security context is complete. At this point, the data can be sent to the server.

7. Normative References

[RFC2743] Linn, J., "Generic Security Service Application Program
           Interface Version 2", 2, Update 1", 2743, January 2000.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
           Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4178] Zhu, L., Leach, P., Jaganathan, K., and W. Ingersoll, "The
           Simple and Protected GSS-API Generic Security Service
           Application Program Interface (GSS-API) Negotiation
           Mechanism", 4178, October 2005.

[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.

[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.

[RFC2712] Medvinsky, A. and M. Hur, "Addition of Kerberos Cipher
           Suites to Transport Layer Security (TLS)", RFC 2712,
           October 1999.

[RFC4121] Zhu, L., Jaganathan, K., and S. Hartman, "The Kerberos
           Version 5 Generic Security Service Application Program
           Interface (GSS-API) Mechanism: Version 2", RFC 4121, July
           2005.

Authors' Addresses

    Karthik Jaganathan
    Microsoft Corporation
    One Microsoft Way
    Redmond, WA 98052
    US

    EMail: karthikj@microsoft.com

    Larry Zhu
    Microsoft Corporation
    One Microsoft Way
    Redmond, WA 98052
    US

    EMail: lzhu@microsoft.com

    John Brezak
    Microsoft Corporation
    One Microsoft Way
    Redmond, WA 98052
    US

    EMail: jbrezak@microsoft.com

Full Copyright Statement

Intellectual Property

Acknowledgement