HTTP-Based Cross-Platform Authentication by Using the Negotiate Protocol

Sanj Surati & Michael Muckin
Microsoft Consulting Services

December 2002

Applies to:
Windows® 2000
Non-Windows web servers
Kerberos-capable clients

Summary: Learn how you can implement Kerberos-based authentication by using HTTP in a cross-platform environment. This article is the first in a series.

The series consists of three parts:

- "Network Infrastructure"–Describes the infrastructure required to enable the solution.
- "SPNEGO Tokens and the Negotiate Protocol"–Describes the negotiate protocol and binary layouts of information sent over the wire.
- "SPNEGO Token Handler API"–Describes and provides the C source code for an API that will parse and create SPNEGO tokens.

Contents

Introduction

With the availability of Kerberos-based authentication in Windows® 2000, there has always been the possibility of interoperable cross-platform authentication with web browsers. Just as integrated authentication in IIS does not require HTTP-based Windows clients to manually re-authenticate themselves by forcing users to reenter their passwords, so can the same functionality be available when communicating with non-Windows web servers from Kerberos-capable clients. The ability to have cross-platform authentication on a network significantly reduces complexity both for administrators and users. With cross-platform authentication, administrators need only worry about Active Directory. Also, users no longer need to remember multiple accounts and passwords, or reenter their credentials when they want to access non-Windows web-based resources.

On a Windows 2000 server with integrated authentication turned on, IIS authenticates with Windows 2000 Internet Explorer clients using what is called the Negotiate authentication package (that implements the SPNEGO protocol). This article will provide a high-level overview of the protocol and show the configuration settings necessary to enable Kerberos as the negotiated authentication mechanism.

Assumptions

Intranet web-based applications

This solution assumes the following to be true:

- The use of this cross-platform authentication solution is intended for *intranet* apps only.
- Microsoft's Active Directory/Kerberos implementation is the single user and authentication store.
- MIT V5 Kerberos is implemented on the UNIX hosts.
- This solution was verified on Solaris 2.8.

This article does NOT cover:

- Custom Kerberos error handling

- Ticket/Session expiration handling

This article assumes that the reader is familiar with Kerberos, the HTTP protocol and C. For an overview of Kerberos authentication in Windows 2000, as well as definitions of important Kerberos-related terminology such as Key Distribution Center (KDC), Ticket Granting Service (TGS), keytab files and more, see Windows 2000 Kerberos Authentication. Additional resources are outlined in Appendix A.

Although this article describes a general cross-platform solution, for ease of reference we will assume non-Windows 2000 web servers to be running a flavor of UNIX with MIT Kerberos V5, which is the environment used to develop this solution. Additionally, although we reference Windows 2000 in this article, the same information is applicable to later versions of Windows (for example, Windows 7 and Windows Server 2008 R2).

Network Topology

Interoperability

There have been many documented and successful implementations of Windows UNIX Kerberos interoperability using the MIT V5 Kerberos standard. The table below describes the typical scenarios encountered with Kerberos interoperability.

Table 1. Kerberos Interoperability Scenarios

	Access to Windows® 2000 resources	**Access to non-Windows 2000 resources**
Windows 2000 client authentication to Windows KDC	Native	One-way Trust or Service Account
Windows 2000 client authentication to non-Windows KDC	One-way Trust or Service Account	Client Configuration
Non-Windows client authentication to non-Windows KDC	One-way Trust or Service Account	Native
Non-Windows client authentication to Windows KDC	Client Configuration	One-way Trust or Service Account

This particular solution is slightly different, however, in that it does not neatly fall into any of the above scenarios; most of these scenarios assume another non-Windows Kerberos realm. In this case, there is no existing Kerberos realm–and one of the main objectives is to get the UNIX servers to be recognized as members of the Windows 2000 Kerberos realm (domain). How this is accomplished is discussed in the "Kerberos Infrastructure Configuration" portion of this article.

System Requirements

To enable a properly functioning environment for HTTP-based cross-platform authentication, the following components are required:

- Server systems:
    - Windows 2000 or later Active Directory
    - Service accounts for mapping Kerberos services
    - Service Principal Names (SPN) configured correctly
    - Key tab files created and exported
- UNIX web servers:
    - MIT Kerberos V5 Generic Security Service API (GSS-API)
    - Web server software configured to use Active Directory Kerberos realm
    - Keytab import and configuration; successful authentication to Kerberos realm
    - Custom SPNEGO code and custom code in web server to perform HTTP header exchanges
- Client systems:
    - Windows 2000 Professional with Service Pack 2 or later Windows client operating system

- Internet Explorer 6.0 with Service Pack 1 or later (on Windows 2000)
- Proper configuration of the browser

Note If you are running Internet Explorer 6.0, please refer to the "Client Side – Internet Explorer" section of this document for specific details about how to configure this version of Internet Explorer.

HTTP-Based Cross-Platform Authentication Overview

The cross-platform authentication design emulates the "Negotiate" behavior of native Windows-to-Windows authentication services on UNIX web servers. For informational purposes, the native Windows Negotiate (also known as Simple and Protected Negotiate (SPNEGO)) sequence is described in the figure below. The Security Support Provider Interface (SSPI) is conceptually equivalent to where GSS-API calls are handled.

Negotiate behavior in Windows 2000

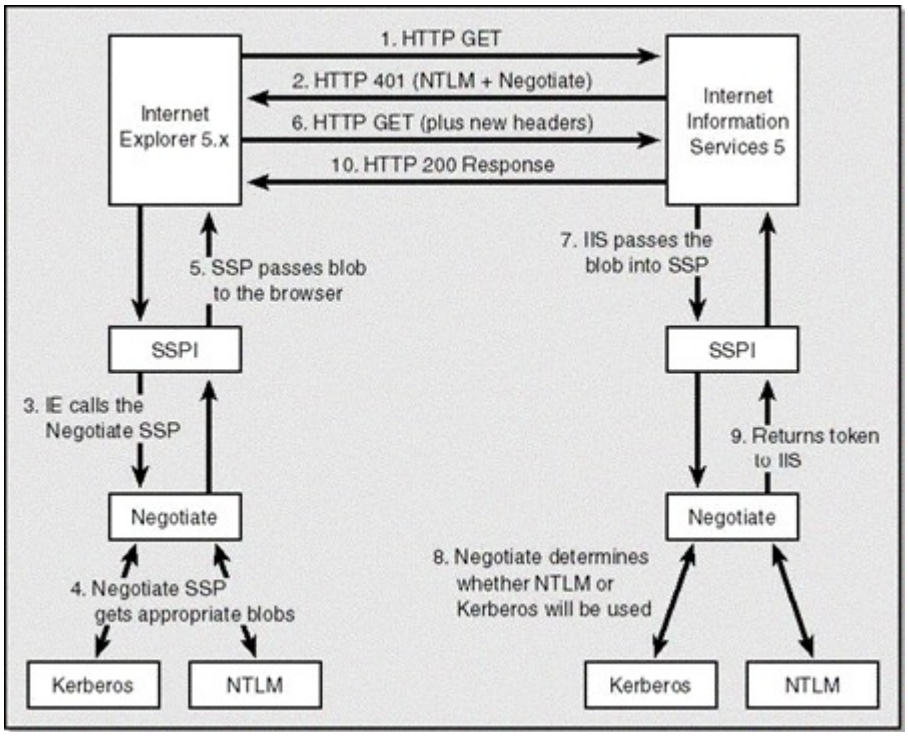

Figure 1. Windows Negotiate Protocol
This figure was taken from *Designing Secure Web-based Applications with Windows 2000* **by Michael Howard and published by Microsoft Press.**

HTTP-based cross-platform authentication design

The design begins with a user who is logged on to the Windows 2000 domain. This is essential–the user must have acquired Kerberos credentials from the domain; local logons will not work.

The logged-on user then attempts to access a web application running on a UNIX web server. The web server asks for authentication with Kerberos (by using SPNEGO). The client obtains a ticket for the requested service from the KDC and presents these credentials back to the web server. The web server extracts–by using the GSS API–the user's credentials and authenticates the request.

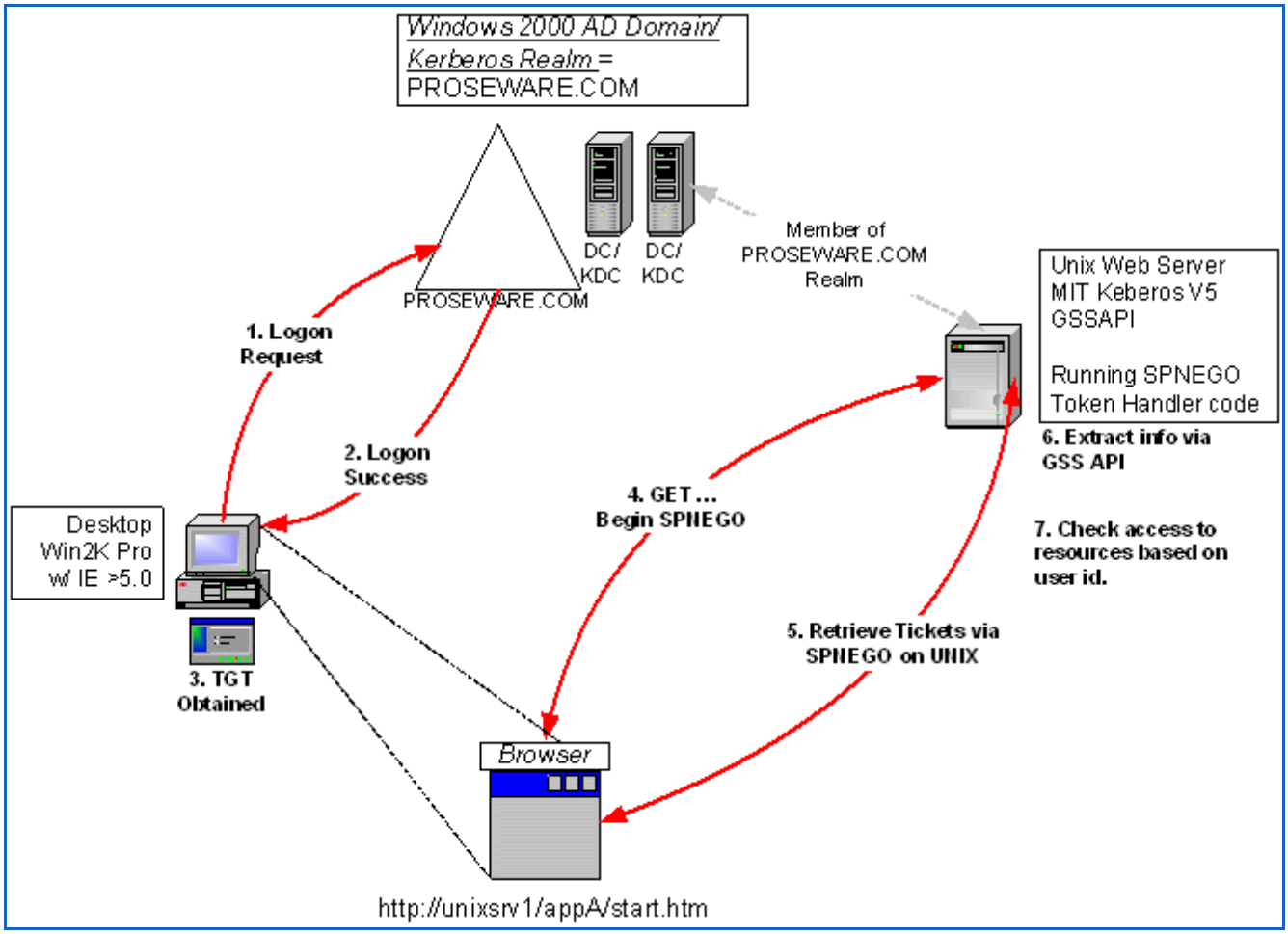

Figure 2. HTTP-based cross-platform authentication overview

HTTP-based cross-platform authentication detail

The following list of steps is a detailed breakdown of the cross-platform authentication design shown above. It elaborates on the topics of process flow of the solution and configuration settings on both Active Directory and the UNIX servers as relating to Kerberos.

1. When the logged-on user requests a resource from the web server, it sends the initial HTTP GET verb.
2. The web server, running the SPNEGO Token Handler code, requires authentication and issues a 401 Access Denied, WWW-Authenticate: Negotiate response.
3. The client calls `AcquireCredentialsHandle()` and `InitializeSecurityContext()` with the SPN to build the Security Context that requests the session ticket from the TGS/KDC.
4. The TGS/KDC supplies the client with the necessary Kerberos Ticket (assuming the client is authorized) wrapped in a SPNEGO Token.
5. The client re-sends the HTTP GET request + the Negotiate SPNEGO Token in an Authorization: Negotiate base64(token) header.
6. The web server's SPNEGO Token Handler code accepts and processes the token through GSS API, authenticates the user and responds with the requested URL.

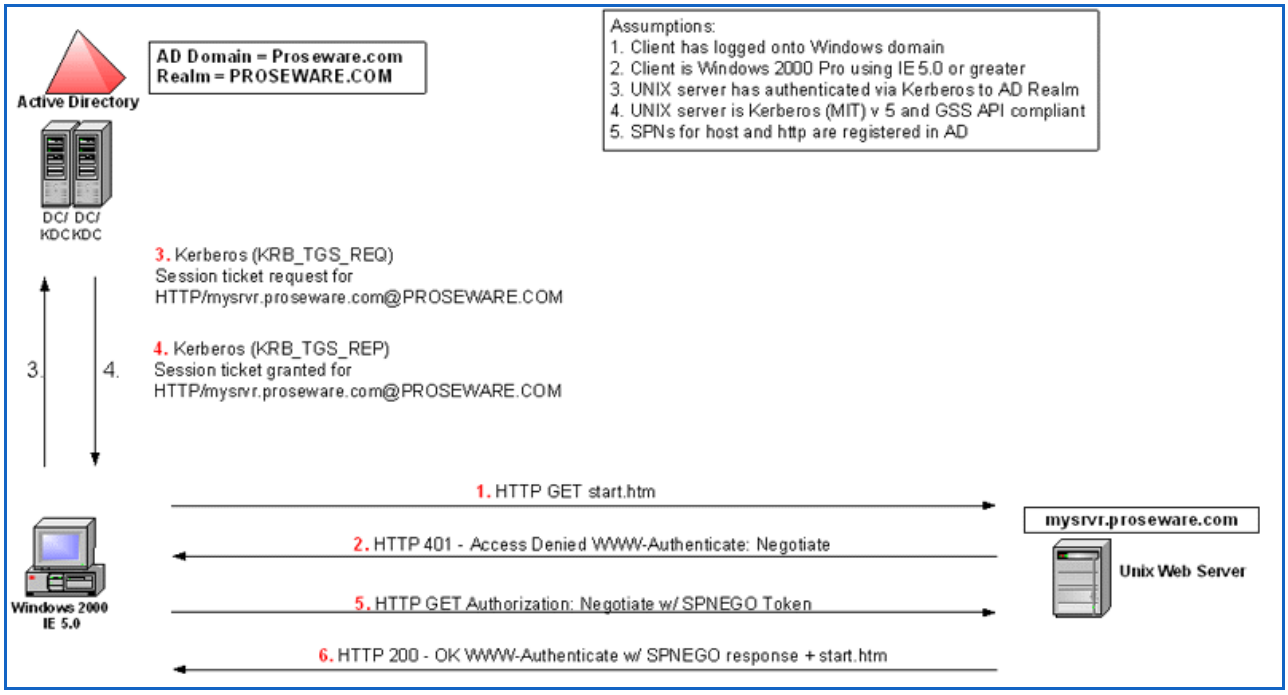

Figure 3. Protocol flow of cross-platform authentication solution

Kerberos Infrastructure Configuration

Now that you have been given a high-level description of the Negotiate Protocol and the Network Topology, you are ready for more detail. This section shows you how to configure your network. Following are the actual machine configuration and settings that must be applied to UNIX hosts, Active Directory and client workstations to reproduce the previously described design.

Windows 2000 and Active Directory

1. Create a User Account in Active Directory for each UNIX host and service.
2. Create the SPNs associated with each User Account–this must be done on a DC.
3. Generate the keytab files for each service.
4. Copy the keytab files to the UNIX hosts.
5. Authenticate the UNIX hosts to the Kerberos realm.

Commands for SPNs, account mappings and keytab files

1. Create the user account in AD using Active Directory Users and Computers Snap-in and set the password. Do NOT select "User must change password at next logon" (remember the password). The account does NOT need to be xxxx.proseware.com–just the host name will do. Example: use mysrvr (not mysrvr.proseware.com)
2. Create the SPNs associated with this account on the KDC:

```
setspn-A host/mysrvr.proseware.com mysrvr
setspn-A HTTP/mysrvr.proseware.com mysrvr
```

> **Note** Use upper-case "HTTP" to match the way Internet Explorer builds SPNs. Alternatively, you can run ktpass (as shown below) to create the SPNs. When using the "**-princ**" option of ktpass, you are specifying the Kerberos principal that is to be registered in the realm; the "**-mapuser**" option should be the account created in step 1 (above). Whether or not you run setspn, ktpass must be run to generate the necessary keytab files.

3. Create and export the keytabs:

```
ktpass -princ host/mysrvr.proseware.com@PROSEWARE.COM -pass
<password> -mapuser mysrvr -out c:\temp\mysrvr.host.keytab
```

```
ktpass -princ HTTP/mysrvr.proseware.com@PROSEWARE.COM -pass
<password> -mapuser mysrvr -out c:\temp\mysrvr.HTTP.keytab
```

UNIX Hosts

1. Log on as **root**
2. Configure the **Krb5.conf** file, as in the following sample:

```
[logging]
 default = FILE:/var/log/krb5libs.log
 kdc = FILE:/var/log/krb5kdc.log
 admin_server = FILE:/var/log/kadmind.log

[libdefaults]
 ticket_lifetime = 24000
 default_realm = PROSEWARE.COM

[realms]
 PROSEWARE.COM = {
  kdc = mysrvr.proseware.com:88
  admin_server = mysrvr.proseware.com:749
  default_domain = proseware.com
 }

[domain_realm]
 .corp.proseware.com = PROSEWARE.COM
 corp.proseware.com = PROSEWARE.COM
 proseware.com = PROSEWARE.COM

[kdc]
 profile = /usr/local/var/krb5kdc/kdc.conf

[pam]
 debug = false
 ticket_lifetime = 36000
 renew_lifetime = 36000
 forwardable = true
 krb4_convert = false
```

> **Note** Solaris hosts may use the "qop" (Quality of Protection) and "mech" files to configure authentication mechanisms in lieu of the [pam] section.

3. After creating the necessary keytab files on the Windows 2000 KDC, securely copy the keytab files to the UNIX host
4. The keytab file for the "host" service should have permissions 600 (r-x------) owned by root. The keytab file for the HTTP should be placed in a directory related to the web server and have appropriate permissions set and the owner set to the web server uid.

Test authentication of the UNIX servers to the Kerberos realm

The next steps will create or merge the keys into a keytab file. A *keytab file* is a list of keys and principals that exist on a host that is using Kerberos authentication. The keys are analogous to passwords. The tool used to accomplish this is the **ktutil** command. **Ktutil** will read the principal/key combinations from the previously generated keytab file and add them as principals in the keytab file. The keytab file should be considered extremely sensitive, and only root access should be given to this file.

1. Run the **ktutil** utility.
2. At the **ktutil:** prompt, type in **rkt** <*PathToKeytabFiles*>**/file.keytab**.
3. Repeat this step for all keytab files.
4. At the **ktutil:** prompt, type in **wkt /etc/krb5.keytab**.

5. At the **ktutil:** prompt, type in **q**.
6. Run the **kinit** tool as follows: **kinit –t /etc/krb5.keytab** <*Host/PrincipalName*>.
7. Enter <*Password*> when prompted.
8. If you are successful, the prompt will be returned; if unsuccessful, an error will be returned to the console.
9. You can verify that authentication worked by using **klist**–to see the TGT for the principal.

Note You can use NetMon on Windows 2000 Server to see the Kerberos traffic (look for UDP traffic or port 88).

Client Side–Internet Explorer

This section will describe how to properly configure the client side of this HTTP cross-platform authentication solution. The client in this case is a web browser, Internet Explorer version 5.0 running on Windows 2000 or later. The combination of Internet Explorer 6.0 running on Windows 2000 requires a unique configuration setting, which is described at the end of this section.

Configure local intranet sites

Because this solution assumes an intranet environment–defined as not exposed to the public Internet–we will next configure Internet Explorer to work with the SSO solution. First, configure the "Local intranet" properties in Internet Explorer.

1. In Internet Explorer, click **Tools**, and then click **Internet Options**.
2. Click the **Security** tab.
3. Click **Local intranet**.
4. Click **Sites**.

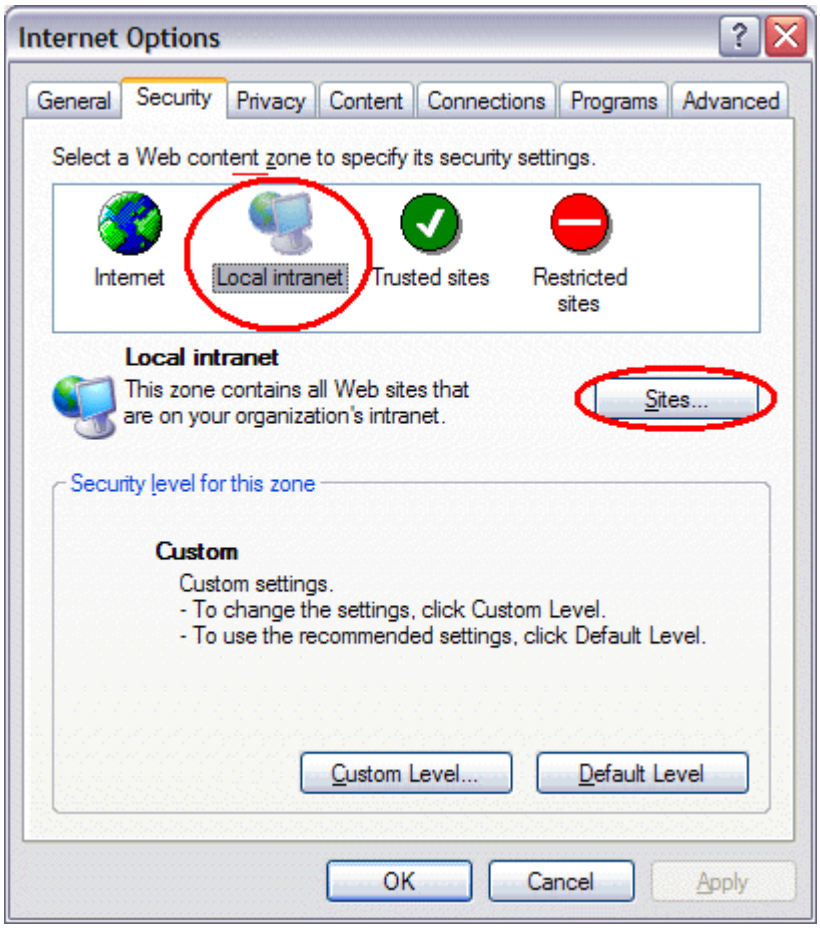

Figure 4. Internet Explorer Internet Options dialog box–Security tab

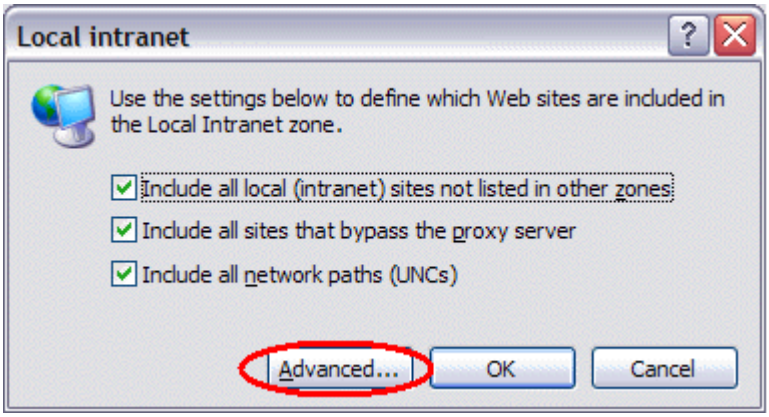

Figure 5. Local intranet dialog box

5. Ensure that **Include all sites that bypass the proxy server** is checked, then click **Advanced**.
6. In the **Local intranet (Advanced)** dialog box, enter all relative domain names that will be used on the intranet (for example, *.proseware.com).

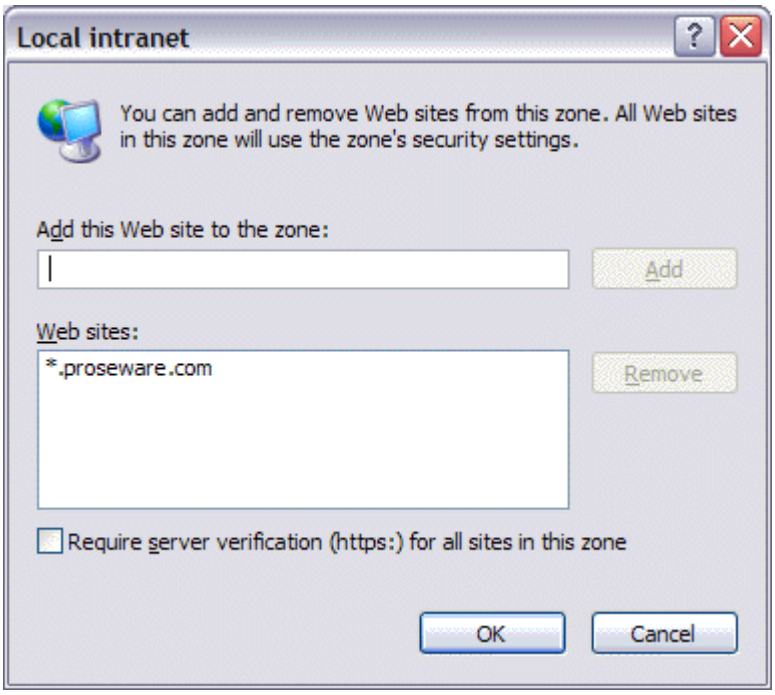

Figure 6. Local intranet dialog box (Advanced)

7. Click **OK** to close the dialog boxes.

Configure intranet authentication

1. Next, click the **Security** tab, click **Local intranet**, and then Click **Custom Level**.

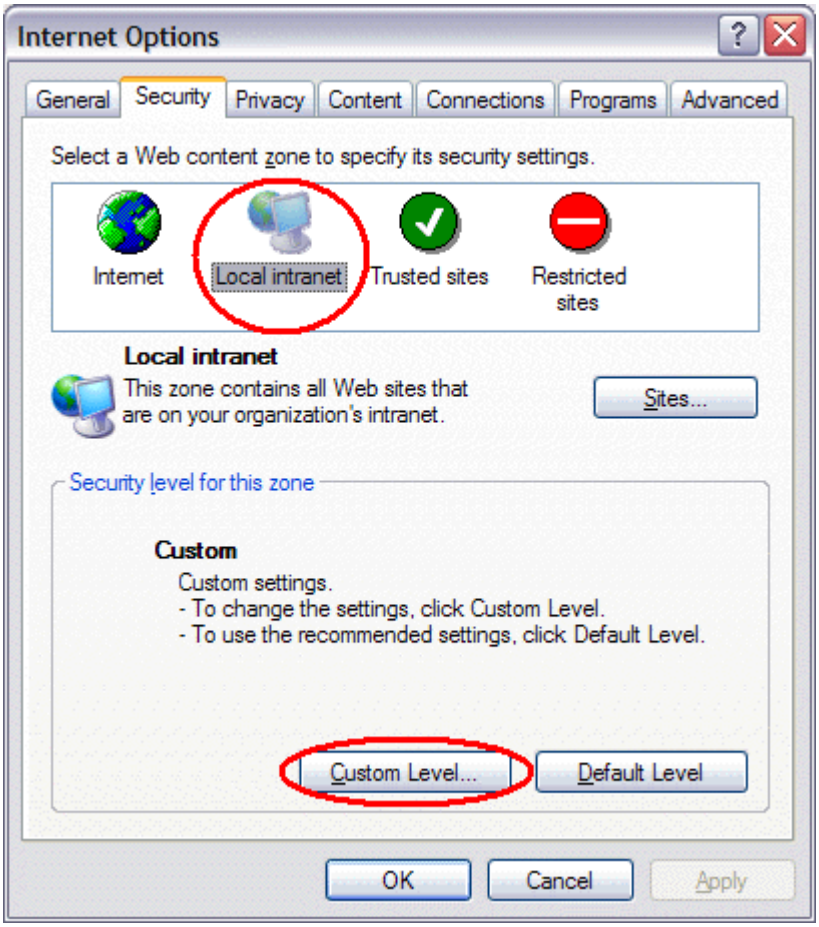

Figure 7. Internet Options dialog box–Security tab

2. In the **Security Settings** dialog box, scroll down to the User Authentication section of the list.

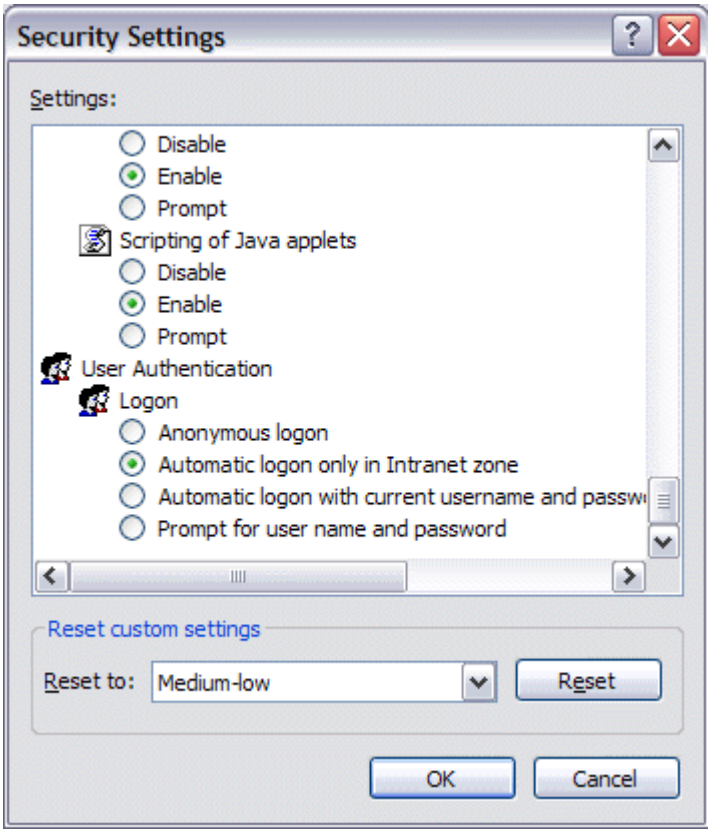

Figure 8. Internet Explorer Security Settings dialog box

3. Select **Automatic logon only in Intranet zone**. This setting will prevent users from having to re-enter logon credentials; a key piece of the solution.
4. Click **OK** to close the **Security Settings** dialog box.

Verify proxy settings

1. In Internet Explorer, click **Tools**, and then click **Internet Options**.
2. Click the **Connections** tab.

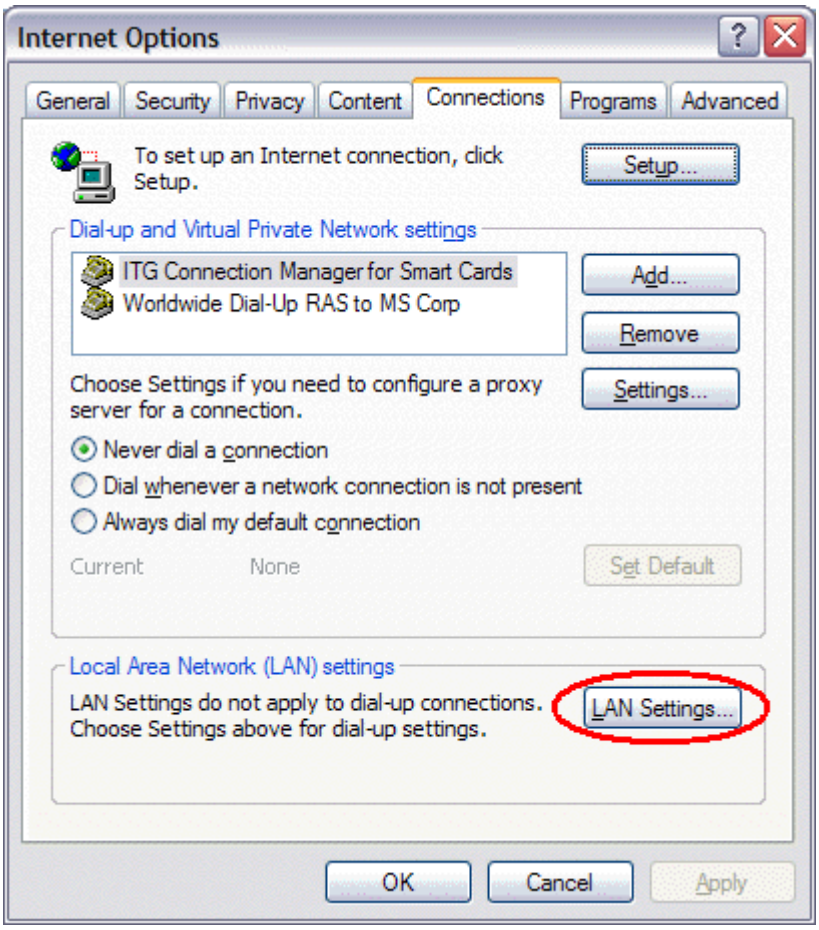

Figure 9. Internet Explorer Internet Options dialog box–Connections tab

3. Click **LAN Settings**.

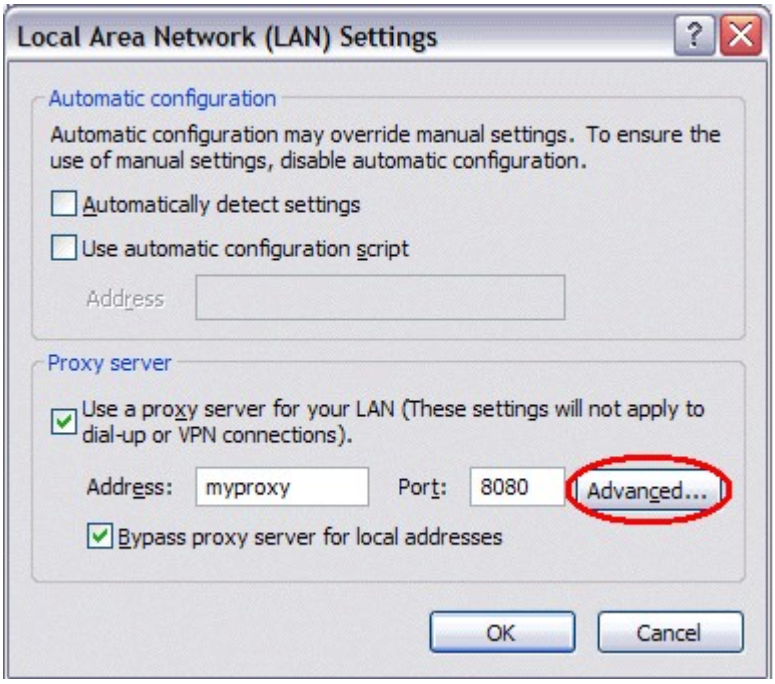

Figure 10. Internet Explorer LAN Settings dialog box

> **Note** An automatic configuration script can be used to create these settings. We are making the settings manually for ease of explanation. For information about using automatic configuration scripts, see Microsoft's TechNet site.

4. Verify that the proxy server address and port number are correct.
5. Click **Advanced**.
6. In the **Proxy Settings** dialog box, ensure that all desired domain names are entered in the **Exceptions** field (for example, *.proseware.com):

Figure 11. Internet Explorer Proxy Settings dialog box

7. Click **OK** to close the **Proxy Settings** dialog box.

Internet Explorer 6.0 Configuration Settings

In addition to the settings made above, one additional setting is required if you are running Internet Explorer 6.0.

1. In Internet Explorer, click **Tools**, and then click **Internet Options**.
2. Click the **Advanced** tab.
3. Scroll down to the Security section.
4. Make sure that **Enable Integrated Windows Authentication (requires restart)** is checked, and then click **OK**.

   If this box was not checked, restart the computer.

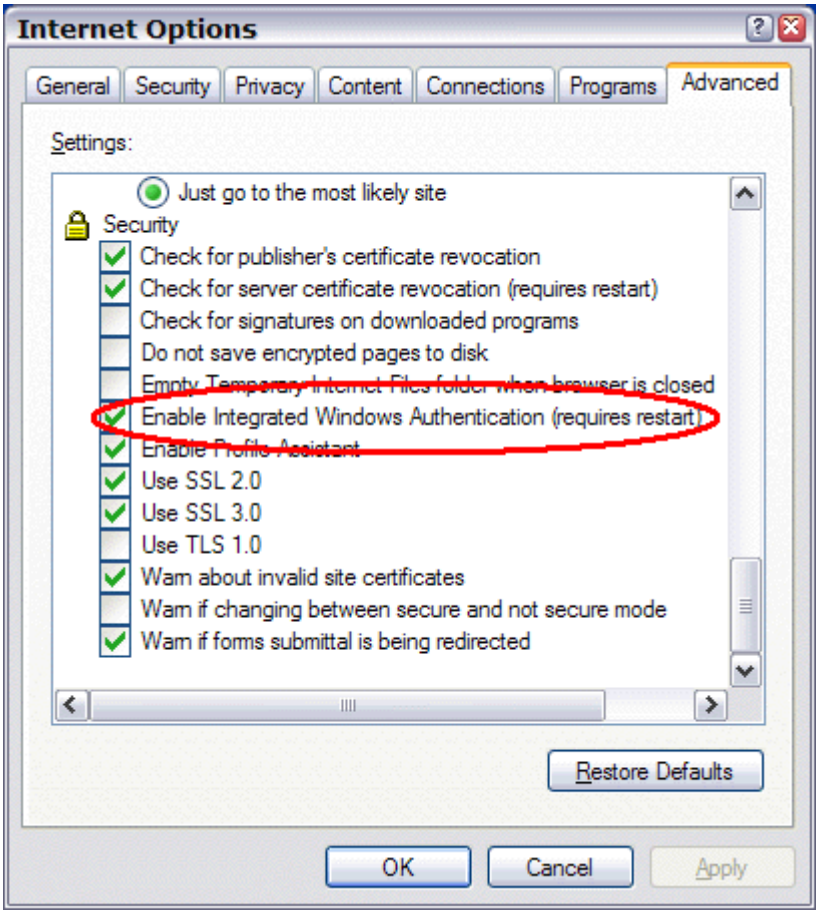

Figure 12. Internet Explorer 6.0 Advanced Security Settings dialog box

Summary

HTTP-based cross-platform authentication is a very powerful capability. As said earlier, it can reduce the burden on both administrators and users. Now that you have an understanding of the required network infrastructure, the following articles in the series, "SPNEGO Tokens and the Negotiate Protocol" and "SPNEGO Token Handler API", will describe binary layouts and provide source code that will assist you in implementing the software end of this solution.

Appendix A–References

- RFC 1510 "The Kerberos Network Authentication Service (V5)"
- RFC 1508 "Generic Security Service Application Program Interface"
- RFC 1964 "The Kerberos Version 5 GSS-API Mechanism"
- RFC 2078 "Generic Security Service Application Program Interface, Version 2"
- RFC 2478 "Simple and Protected GSS-API Negotiation Mechanism"
- RFC 4559 "SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows"
- MIT Kerberos Web Site
- Microsoft Kerberos Links:
    - Windows Server, search for "Basic Kerberos."
    - Step-by-Step Guide to Kerberos 5 (krb5 1.0) Interoperability
    - How a Service Composes its SPNs