Paper 312-2010

Integrated Windows Authentication Support for SAS® 9.2 Enterprise BI Web Applications

Heesun Park, SAS Institute Inc., Cary, NC

ABSTRACT

Integrated Windows Authentication (IWA) support for Web applications provides a Single Sign-On solution in the Windows environment. After examining the Kerberos and Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) protocols, which are the backbones of IWA support for Web applications, this paper explains IWA configuration options for SAS Enterprise BI (EBI) Web applications. There is a significant difference in IWA configuration for SAS 9.1.3 and SAS 9.2. The new logon process for SAS 9.2 Enterprise BI Web applications requires advanced setup for IWA/SPNEGO support in the application servers. This paper presents the major steps involved in configuring support for SPNEGO in the application servers along with information about configuring SAS EBI Web applications with Web authentication in an IWA environment.

Introduction

The Windows operating systems currently provide two protocols for user authentication. These protocols include the proprietary Windows NT LAN Manager (NTLM) protocol from Microsoft, and the open-source Kerberos protocol developed at Massachusetts Institute of Technology. Beginning with Windows 2000, Kerberos has become a primary protocol for Windows' networks. Kerberos is considered more secure and reliable than NTLM. Also, it can be implemented in any type of network. Integrated Windows Authentication (IWA) is not a protocol. It is a term that is commonly used to refer to an authenticated connection between a client in the Windows domain and other servers or applications located in or outside of the Windows domain through Active Directory. IWA might include the Microsoft NTLM protocol, but in this paper, we focus on the Kerberos protocol, because it is the only protocol supported by Web application servers such as WebSphere 6.x (and later) and WebLogic 9.x (and later).

Beauty of Kerberos Protocol

In order to understand the Kerberos protocol, we need to learn about a few components and terms. In a Kerberos environment, the Key Distribution Center (KDC) is similar to a Domain Controller. It is the central security authority that contains information about all principals and their passwords. The KDC contains two major services: the Authenticating Service (AS) and Ticket Granting Service (TGS). The Kerberos ticket is an important concept for the Kerberos protocol. The Kerberos ticket is issued by KDC and contains the data structure that allows the client to be authenticated in the network. The following fields play a major role in the Kerberos ticket operation:

- Service Principal Name (SPN) or "Server" name (clear text)
- User Name / Domain (encrypted with SPN's password)
- Session Key (encrypted with SPN's password)
- Time Stamp (encrypted with SPN's password)

The Server Principal Name (SPN) contains the name of the server with which it might be used. The remaining fields in the ticket are encrypted with the SPN's password. This means that only the intended server can decode the service ticket and grab the session key for subsequent authentication of the client and communication with the client. The session key is created by the KDC and is encrypted with the client's password before it is sent to the client. Figure 1 shows the ticket and the session key received from the TGS:

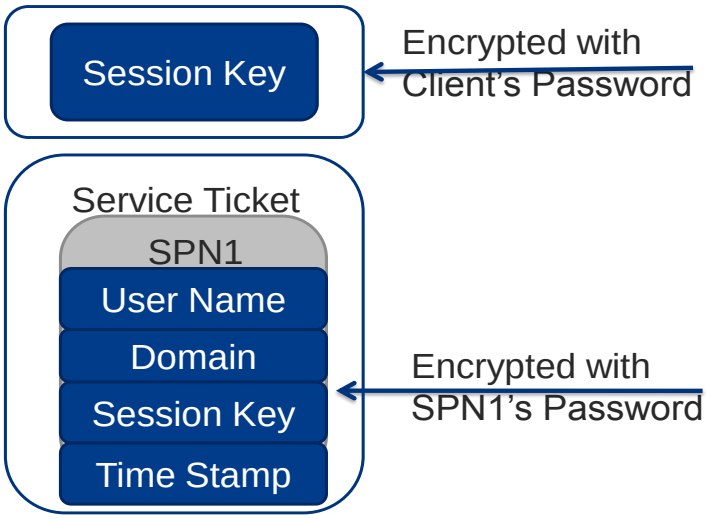

Figure 1: Kerberos Ticket Received from TGS

Another important data structure maintained by the client is an authenticator. An authenticator, which consists of the client's identification and time stamp, is created by the client and encrypted with the session key that is shared between the specific server and the client. Kerberos is a mutual-authentication protocol. The authenticator, which is constructed and sent by the client, identifies the client. The authenticator is encrypted with the session key, therefore only the intended server can decrypt the authenticator and verify the client's identity. Figure 2 depicts the service ticket and the creation of the authenticator ~~using~~ with the session key. Both the ticket and the authenticator are sent to the server.

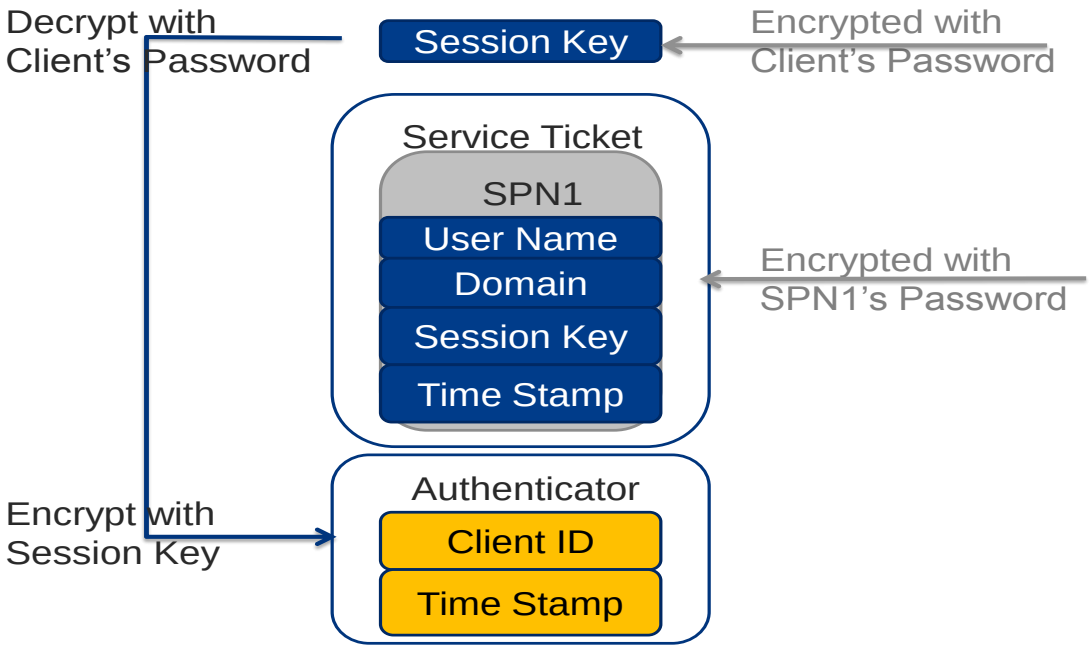

Figure 2 Authenticator construction on client side

Kerberos authentication consists of three main phases. In the first phase, the client directly contacts the AS in the KDC and requests a ticket-granting ticket (TGT). This TGT allows the client to communicate with the TGS in order to obtain service tickets for use with other servers. Upon receiving this initial request, the AS creates a new session key that the client and TGS can use for cryptographic communication sessions. The encryption algorithm for the session key is selected from the list defined in the Kerberos configuration. This session key is never reused. Two copies of this session key are created. One copy is placed in the TGT and encrypted with the TGS's password. The other copy is encrypted with the client's password. The AS has the ability to create the new session key because it has access to all user names and the associated passwords. Along with the TGT, the AS sends the session key (encrypted with the client's password) back to the client. If the client's identity is correct, the client can decrypt the session key for use with the TGS.

In the second phase of Kerberos authentication, the client uses the TGT to obtain a service ticket for the target server (SPN). The client sends a request to the TGS with a copy of the TGT and an authenticator. . Upon receiving this request, the TGS first opens the ticket using its own password, and it. extracts the new session key. Using the session key, the TGS can decrypt the authenticator. If all goes well, the TGS now knows with whom it is communicating. By looking at the timestamp, it can also verify that it is a recent request, and that it is not an attempted replay attack by an intruder. The TGS creates a new session key for the target server and the client. It encrypts one copy of the session key with the client's password. The other copy is placed into a new service ticket for use with the requested server. The new session key and ticket are returned to the client.

In the final stage, the client decrypts the session key for use with the server and encrypts the authenticator. It sends a request to the server, along with the service ticket that the TGS created. The client also sends an authenticator.. Just as the TGS was able to verify the client's identity, the server can decrypt the ticket with its own password. It extracts the session key from the ticket and uses the session key to decrypt the authenticator. The server and client now possess the same session key, which can be used to encrypt or sign messages.

As an option, the server can authenticate itself to the client by creating an authenticator that is encrypted with the session key, and sending it to the client. Since the server is the only other principal that can possess the session key, the server's identity is proven to the client. Figure 3 shows the Kerberos authentication process.

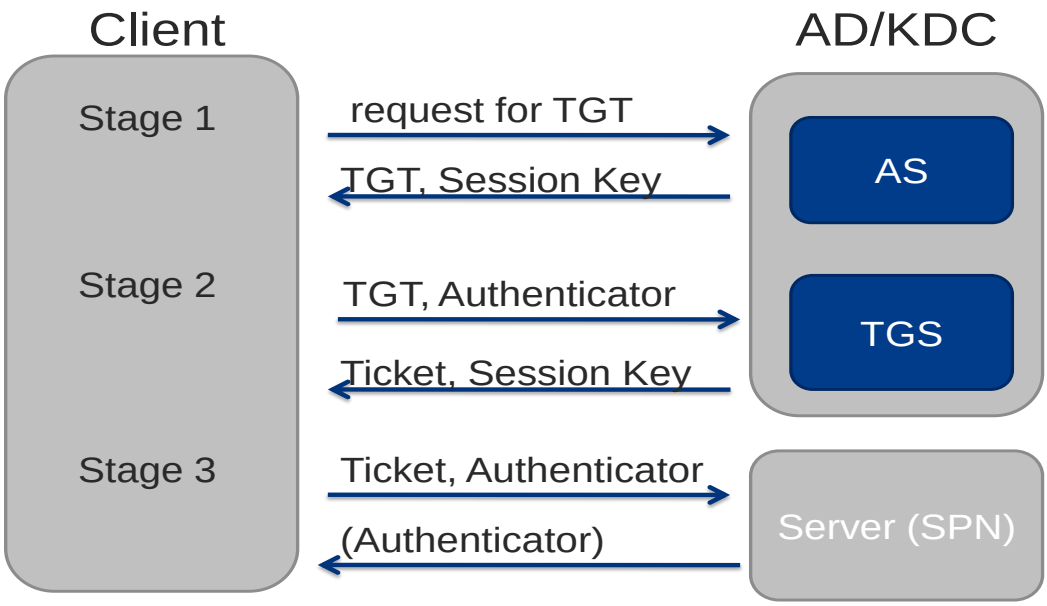

Figure 3: Kerberos authentication process

Notice that the "Server" or "SPN" in this paper represents a Web application server such as WebSphere, WebLogic, or JBoss, which host the SAS 9.2 EBI Web applications. In an IWA configuration, the Web application server can reside either within or outside of the Windows domain, but the client should reside within the Windows domain. The Web application server becomes a Kerberos entity when you define it as a user in the Active Directory and map the user to the Web application server (using the **setspn** command). The Web application server (SPN) is represented by the user in the KDC and uses its password to decode the incoming authenticator. A keytab file that contains the

user's password should be created for the Web application server from the KDC and copied to the machine where the application server is running. The keytab file becomes a part of Kerberos configuration for the Web application server.

SPNEGO, NTLM, and Kerberos

SPNEGO is based on the Generic Security Service Application Programming Interface (GSSAPI), which provides a compatible interface to applications and shields them from the underlying security service or protocol. SPNEGO was first developed in Internet Explorer 5 and IIS 5 to provide Single Sign-On in a Windows environment. Although SPNEGO is an authentication protocol, it does not perform any authentication. Instead, it negotiates whether the client and server will use the NTLM or Kerberos protocols. If both client and server support the Kerberos protocol, then Kerberos is used. Otherwise, the client and server will attempt to use NTLM. NTLM is not supported by most Web application servers, therefore both the client and server side support Kerberos through SPNEGO.

JAAS Configuration for SPNEGO

When a Web application server becomes a Kerberos entity (SPN), it should be configured to handle incoming SPNEGO traffic. The Java Authentication and Authorization Service (JAAS) [4] is the backbone of the authentication process for Web application servers and Web applications. JAAS, in essence, is a Java implementation of PAM (Pluggable Authentication Module), and in its simplest form it is like Java Database Connectivity (JDBC), an abstraction over authentication module providers. To a large extent, it enables Web applications to be independent of the authentication method, and thus enables them to become portable.

A JAAS implementation consists of a stack of JAAS login modules that handle different types of authentication methods. For example, a Web application server carries JAAS login modules for basic authentication, certificate-based authentication, token-based authentication, and so forth. Native SPNEGO support from the application server means that a JAAS login module that handles an SPNEGO token can be configured for the Web application server.

JAAS login modules are defined in the Web application server's login-config.xml (or equivalent) file, but each Web application server provides a different user interface to accomplish this task. An SPNEGO token (which has a Kerberos ticket inside of it) already contains authenticated user information, therefore an SPNEGO login module should be at the top of the JAAS login module stack. WebSphere handles the situation with its Trust Association Interceptor (TAI) utility. When a SPNEGO TAI is set and enabled, it intercepts the incoming SPNEGO token, decodes the Kerberos ticket using the keytab associated with the application server, and places the JAAS principal into the JAAS Subject. Other JAAS login modules can be defined in the configuration, including the SAS 9.2 TrustedLoginModule, which further authenticates the user through the SAS Metadata Server. WebLogic has a similar mechanism known as Identity Asserter. For JBoss, you edit the login-config.xml file in the JBoss server's configuration directory, and add the definition of the SPNEGO login module manually. The following section shows the SPNEGO login module definition in a login-config.xml file for JBoss:

```
<application-policy name="SPNEGO">
 <authentication>

  <login-module code="org.jboss.security.negotiation.spnego.SPNEGOLoginModule" flag="requisite">
   <module-option name="password-stacking">useFirstPass</module-option>
   <module-option name="serverSecurityDomain">host</module-option>
  </login-module>

  <login-module code="org.jboss.security.auth.spi.UsersRolesLoginModule" flag="required">
   <module-option name="password-stacking">useFirstPass</module-option>
   <module-option name="usersProperties">props/spnego-users.properties</module-option>
   <module-option name="rolesProperties">props/spnego-roles.properties</module-option>
  </login-module>

  <!-- Add SASTrustedLoginModule definition here -->

 </authentication>
</application-policy>
```

Another configuration that is tied to user authentication is security role mapping of the incoming user. Security role mapping is a protection mechanism administered through the Web application server. A Web application defines a

role-name in its deployment descriptor (web.xml), and the Web application server delivers the physical users into the role-name group. The implementation is specific to the Web application server. It can be done through properties files for roles and users as in the previous example, through a roles' group definition in an LDAP structure, or through the use of a dynamic role-mapping utility (WebSphere).

Web Application Protection Mechanism

Security implementation for J2EE [5] Web applications consists of two parts. One is authentication, which basically controls access to the Web application itself. The other is authorization, which controls the type of operation allowed on resources (for example, servers and data) by the authenticated user. This paper mainly focuses on the authentication process for Web applications, which requires coordination with an existing Web infrastructure. SAS 9.2 EBI comes with a SAS Metadata Server that provides very sophisticated resource permission control. The goal of SSO is to use an authentication mechanism provided by the enterprise through its user registry and to seamlessly integrate it with the resource authorization service available in the SAS 9.2 Metadata Server. .

J2EE Web applications, including SAS 9.2 BI Web applications, can be protected in two ways. The first method relies upon the Web application to provide its own authentication mechanism. SAS metadata-based authentication, or SAS "host" authentication, falls into this category. This method is efficient for a simple configuration that does not require web perimeter security protection involving an external user registry. The other method delegates the Web application server's authentication Web to itself by using the deployment descriptor (web.xml file). In the deployment descriptor, you can define security role mapping as an authentication method for Web applications. . This mechanism allows the use of many types of authentication methods including SPNEGO, which allows a user logged in to the Windows domain to access a Web application deployed in the Web application server. Security role mapping is another layer of protection for Web application access. It is a two-step process. First, you define the name of a logical group in the <role-name> element within the Web application's deployment descriptor (web.xml). As a result, only users who belong to the group can access the Web application. , Next, the Web application server delivers the physical users to the logical grouping. If the Web application server's administrator can manage the security role mapping dynamically and independently, that is the best method (WebSphere works that way). Other methods include the use of a roles' property file or an LDAP group that matches a role name defined in the Web application server's deployment descriptor. Here is a sample <security-constraint> section of the web.xml file that shows the authentication method and role name:

```
<security-constraint>
    <web-resource-collection>
     <web-resource-name>Sample </web-resource-name>
     <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
     <role-name>SASWebUser</role-name>
    </auth-constraint>
    <user-data-constraint>
     <transport-guarantee>NONE</transport-guarantee>
    </user-data-constraint>
 </security-constraint>

 <login-config>
   <auth-method>SPNEGO</auth-method>
   <realm-name>SPNEGO</realm-name>
 </login-config>

 <security-role>
    <role-name>SASWebUser</role-name>
 </security-role>
```

The designation of the auth-method parameter for IWA is flexible (and is application server specific) because the authenticated user is populated into the JAAS subject before the Web application is invoked. The previous example (<auth-method>SPNEGO</auth-method>) works for JBoss 4.x. However, WebLogic requires <auth-method>CLIENT-CERT</auth-method> for SPNEGO authentication. Consult your Web application server's documentation for instructions on setting up SPNEGO for the auth-method designation as it applies to Web applications. .

IWA Support Configuration Differences between SAS 9.1.3 and SAS 9.2

There are significant differences in the configuration processes for Web authentication in SAS 9.1.3 and SAS 9.2, and this necessitates a basic configuration change for IWA support. In essence, the change applies to how SAS gets and processes the externally-authenticated user name. SAS 9.2 is based on a full JAAS implementation from the Web application server, while SAS 9.1.3 relies on the user information passed in through HTTP header. Figure 4 shows the difference in the authentication path:

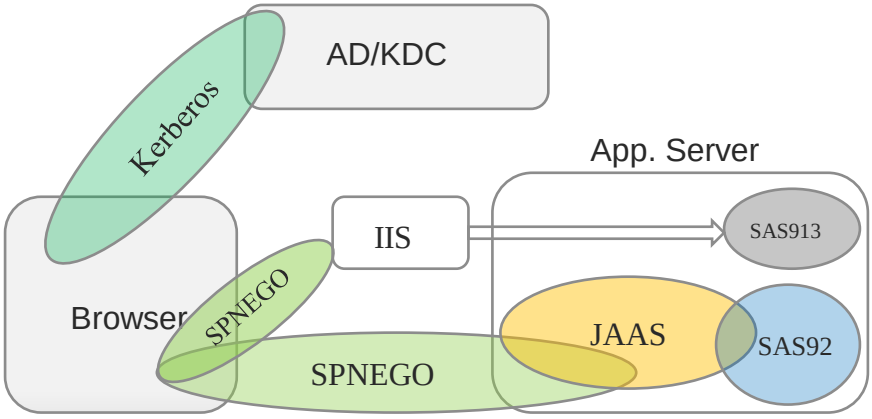

Figure 4: Conceptual IWA Authentication Path for SAS 9.1.3 and SAS 9.2

Clearly, JAAS is the central component in authentication service for Web application servers and is much more secure than other approaches such as use of an HTTP header for the user name. Also, native SPNEGO support from the Web application server makes JAAS-based IWA possible.

SAS 9.1.3 IWA Support through IIS or a Servlet Filter

SAS 9.1.3 Web authentication relies on the user name from the remoteUser field in the HTTP header. This field can be filled in by a reverse proxy server after it authenticates the user. For example, this process occurs after IIS authenticates the user through IWA. Also, the Web application server could place the user name in this field after it runs its JAAS operation. The major purpose of JAAS processing is to add an authenticated user into the JAAS Subject as a JAAS Principal. In the process, it adds the user name to the remoteUser field of the HTTP header as well. The SAS TrustedLoginModule grabs the user name from remoteUser field, and takes it to the SAS Metadata Server for SAS resource authorization. In other words, the SAS 9.1.3 Web authentication process was not a purely JAAS-based operation as it does not directly use JAAS Subjects or Principals inside of it. SAS Web authentication was possible through user authentication by the reverse proxy server, and without going through JAAS processing in the Web application server. This logic enables IWA configuration with IWA based on IIS or Web application servlet filter that is based on IWA . In the SAS 9.1.3 timeframe, major Web application servers, such as WebSphere 5.x, did not support IWA (SPNEGO) natively, so the JAAS-based IWA support from the Web application server was not possible. Figure 5 shows an IWA configuration with IIS and Tomcat. In this case, IIS carries out the user authentication and passes the authenticated user name to Tomcat. In turn, authentication should be turned off from Tomcat so that you bypass any JAAS processing. In SAS 9.1.3, the TrustedLoginModule defined for Web applications directly consumes the user name from the HTTP header for SAS resource authorization through the SAS Metadata Server. For a fully enhanced Web application server such as WebSphere, one cannot typically turn off security, so the IIS based IWA configuration does not work with it.

Another possible scenario for IWA configuration with SAS 9.1.3 is the use of a servlet filter defined for a Web application. In this case, the Web application itself assumes the external user authentication responsibility through its own servlet filter. The servlet filter gets control first when the Web application is invoked. It should be able to handle

incoming SPNEGO token and authenticate the user through Kerberos protocol. The servlet filter implementation is typically a 3rd party product. Like IIS – Tomcat configuration above, authentication should be turned off from Tomcat as the Web application itself is doing the user authentication. This configuration becomes less attractive as the Web application servers start to support SPNEGO natively.

SAS 9.2 IWA Support Through a Web Application Server

In the SAS 9.2 timeframe, two major changes occurred in support of an IWA configuration. First, SPNEGO/Kerberos protocol is supported inside of the Web application server through a JAAS login module. In addition, the SAS 9.2 authentication process is tightly coupled with the Web application server's JAAS operation. With these changes, SAS 9.2 was able to support IWA configuration with WebSphere 6.1The JBoss 4.x Negotiation module was available in early 2009, and the second maintenance release of SAS 9.2 supports IWA with JBoss. IWA support for WebLogic 9 (and 10) is scheduled for the third maintenance release of SAS 9.2.

Let's take a close look at the Web authentication process for SAS 9.2 EBI Web applications. In support of JAAS, SAS supplied a SASTrustedloginModule that should be chained with application-server-supplied login modules. SAS 9.2 EBI Web applications are now protected through a single SAS Web application called the SAS Logon Manager. Each protected SAS 9.2 Web application contains a "security filter" servlet. This servlet redirects the log in request to SAS Logon Manager. SAS Logon Manager is protected by the Web application server through its <security-constraint> section in the deployment descriptor (web.xml). The Web application server runs the JAAS login modules when access to the SAS Logon Manager is initiated. The JAAS login-modules stack, which consumes the JAAS Principal created by the system login modules, consists of Web application server's system login modules and the SAS 9.2 TrustedLoginModule at the bottom. . After successful external and SAS metadata authentication, the SAS Logon Manager creates a SAS user and session context for the original Web application and returns the control to it. The following <application-policy> definition in the JBoss server's login-config.xml file for SASApplicationLogin shows the JAAS login module stack. This JAAS login module stack runs for the user authentication which applies to the SAS Logon Manager Web application:

```
 <application-policy name="SASApplicationLogin">
<authentication>
<login-module code="org.jboss.security.negotiation.spnego.SPNEGOLoginModule" flag="requisite">
  <module-option name="password-stacking">useFirstPass</module-option>
  <module-option name="serverSecurityDomain">host</module-option>
</login-module>

<login-module code="org.jboss.security.auth.spi.UsersRolesLoginModule" flag="required">
    <module-option name="password-stacking">useFirstPass</module-option>
    <module-option name="usersProperties">props/spnego-users.properties</module-option>
    <module-option name="rolesProperties">props/spnego-roles.properties</module-option>
 </login-module>

<login-module code="com.sas.services.security.login.jboss.JBossTrustedLoginModule" flag="required">
  <module-option name="host">sasebi.test.sas.com</module-option>
  <module-option name="port">8561</module-option>
  <module-option name="repository">Foundation</module-option>
  <module-option name="domain">web</module-option>
  <module-option name="trusteduser">sastrust@saspw</module-option>
  <module-option name="trustedpw">encrypted-pw</module-option>
  <module-option name="debug">true</module-option>
</login-module>
</authentication>
</application-policy>
```

At the top is the SPNEGOLoginModule, which handles the SPNEGO token and places a JAAS Principal in the JAAS Subject. The second in line is the UserRolesLoginModules, which checks the security role mapping for the users. Only users defined in the security role for the Web application can access the Web application.

Figure 6 shows the IWA configuration and SPNEGO/Kerberos authentication process for SAS 9.2 EBI Web applications. The Web application server becomes a Kerberos entity (SPN) through user mapping and use of the

keytab file as its JVM (Java Virtual Machine) parameter, therefore it does not have to be inside the Windows domain. This arrangement could provide significant flexibility for application-server configuration and performance management. Client machines, though, should be in the same Windows domain.

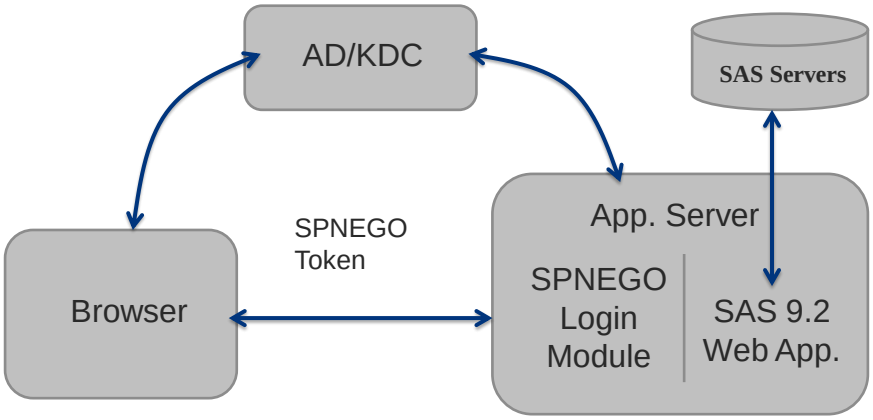

Figure 6: IWA/SPNEGO configuration for SAS 9.2 EBI Web Applications

1. User logs in to the Windows domain by requesting (TGT) from KDC.
2. User receives TGT from Kerberos KDC.
3. User accesses protected SAS Web application from the browser.
4. Web application server sends 401 error with Authentication-Negotiation option
5. Browser requests service ticket for Web application server's SPN
6. Browser receives Service Ticket for Web application server's SPN
7. Browser creates an SPNEGO token that contains Kerberos service ticket and sends it with the request.
8. The SPNEGO login module in the Web application server decodes the SPNEGO token and Kerberos service ticket, and obtains the user name.
9. The SPNEGO login module validates the user with the Web application server's registry (which is Active Directory) and adds the JAAS Principal to the JAAS Subject.
10. The SAS Web application produces output by accessing SAS servers and SAS resources.
11. Output is returned to the browser with session information.

SAS 9.2 EBI Web Application Deployment Process and IWA Configuration

The SPNEGO setup, which occurs between the Web application server and Windows Domain Controller, can be set up independent of any Web application. Mainly, it is the responsibility of the Windows administrator and the Web application server administrator. The definition of the Web application server as an SPN, mapping of a user onto it, and creation of a keytab file for the Web application server require Windows domain administrator privileges. Therefore, it is highly recommended that any organization that plans to implement IWA should first configure and test the SPNEGO setup for the Web application server. This test configuration should be completed before deploying any Web applications. Each Web application server provides documentation [6,7,8] on SPNEGO setup. The handling of the SPNEGO protocol in the Web application server becomes a part of the JAAS authentication process. The logic is the same, but each Web application server has a different user interface and implementation. When the SPNEGO support is configured for the Web application server, make sure that the JAAS Subject contains the Principal created from the user name which was extracted from the SPNEGO token (Kerberos ticket). JBoss supplies the test program called negotiation-tool-kit with its Negotiation (SPNEGO) support module. For WebSphere and WebLogic, you can use the SEC_CON tool provided by SAS to display the content of the JAAS subject. SEC_CON tool is available from SAS technical support for download.

SAS 9.2 EBI Web applications consist of a number of major Web applications that interact with the Web application server's internal services, such as its mail service, message queues, data sources, and other items. They also require many application server JVM parameters that need to be set properly to ensure the best performance of the Web application. It is fairly difficult to deploy SAS 9.2 EBI Web applications manually. Fortunately, the SAS Deployment Wizard automates the configuration of the Web application server and deployment of SAS 92 EBI Web applications. The SAS Deployment Wizard creates an application server instance and deploys the SAS 9.2 EBI web applications to it. By default, SAS Web applications are configured to use local OSSAS authentication (formerly known as "host" authentication). When the SAS Deployment Wizard completes its basic configuration, the application server instance (typically SASServer1) needs to be configured to support the SPNEGO protocol. An independent application server setup exercise for SPNEGO before the SAS 9.2 EBI installation would be helpful. . To take advantage of SPNEGO authentication by the application server, the SAS 9.2 Web application should be converted to trusted Web authentication per the SAS documentation [9, 10,11]. In summary, the following sequence is recommended as the best practice:

1) Set up SPNEGO support for the application server and test the configuration independently.
2) Run SAS Deployment Wizard to create a SAS server instance and deploy the SAS 9.2 Web applications.
3) Configure the SAS server instance to support SPNEGO protocol.
4) Convert the SAS 9.2 Web applications to use Web authentication to consume the Windows domain-authenticated user supplied through the SPNEGO protocol and JAAS subject

CONCLUSION

To properly configure and support SAS 9.2 EBI Web application in an IWA environment, we need to focus on three things. First, a basic understanding of the Kerberos protocol and Windows domain controller operation is required. The Kerberos protocol is very sophisticated and complex. Deeper understanding of the protocol is a great asset that helps in debugging and resolving problems. The main component of the Windows domain controller is the Active Directory. It contains user definitions in an LDAP structure, but the administration interface has its own flavors. In some cases, a Web application security role name needs to be defined in the Active Directory for security role mapping. Second, it is critical to understand the JAAS configuration inside the Web application server and the Web application protection mechanism. SPNEGO support is implemented through a JAAS login module that initializes a JAAS Subject with a Principal. The JAAS Principal in the Subject is consumed by the TrustedLoginModule that is supplied by SAS. , Finally, knowledge of the authentication logic for the SAS 9.2 EBI Web applications is important to integrate and take advantage of SPNEGO authentication. The main task is to update the deployment descriptor (web.xml) of the SAS Logon Manager Web application. The deployment descriptor tags on to the user authentication provided by the application server, and adds entries for the externally-authenticated users in the SAS metadata .

REFERENCES

Integrated Windows Authentication
http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/523ae943-5e6a-4200-9103-9808baa00157.mspx?mfr=true

Simple and Protected GSS-API Negotiation Protocol (SPNEGO)
http://tools.ietf.org/html/rfc4178

Kerberos: The Network Authentication Protocol
http://web.mit.edu/Kerberos/

Java Authentication and Authorization Service (JAAS) Reference Guide
http://java.sun.com/j2se/1.4.2/docs/guide/security/jaas/JAASRefGuide.html

J2EE Specification, Chapter 3 Security
http://java.sun.com/j2ee/j2ee-1_4-fr-spec.pdf

SPNENO TAI Configuration (for WebSphere)
http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/rsec_SPNEGO_tai_reqs.html

Enabling SPNEGO based Single Sign-On (for WebLogic)
http://download.oracle.com/docs/cd/E13169_01/ales/docs26/integrateappenviron/spnego.html

JBoss Community. “User Guide for JBoss Negotiation (inside of Negotiation-2.0.3.GA module),” Available at https://jira.jboss.org/jira/browse/SECURITY-343

SAS Institute Inc. (2009), “Configuring IBM WebSphere Application Server 6.1 for Web Authentication with SAS 9.2 Web Applications,” http://support.sas.com/resources/thirdpartysupport/v92/appservers/ConfiguringWASWebAuth.pdf

SAS Institute Inc. (2009), “Configuring Oracle WebLogic Server 9.2 for Web Authentication with SAS 9.2 Web Applications,” Available at http://support.sas.com/resources/thirdpartysupport/v92/appservers/ConfiguringWLSWebAuth.pdf

SAS Institute Inc. (2009), “Configuring JBoss Application Server 4.2.0 for Web Authentication with SAS 9.2 Web Applications,” Available at http://support.sas.com/resources/thirdpartysupport/v92/appservers/ConfiguringJBossWebAuth.pdf

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Heesun Park
SAS Institute Inc.
E-mail: sashsp@sas.com