Paper 481-2013

Hardening a SAS® Installation on a multi tier installation on Linux

Jan Bigalke, Allianz Managed Operations & Services SE

ABSTRACT

The security requirements of today require in some use cases the hardening of a SAS® Installation. This paper describes the practical steps of securing the SAS web applications and the impact to the Base SAS® Services on the SAS compute tiers. The SAS Enterprise BI Server will be the object of this explanation. The principals of a secure architecture will be described and the options to secure the individual components presented.

INTRODUCTION

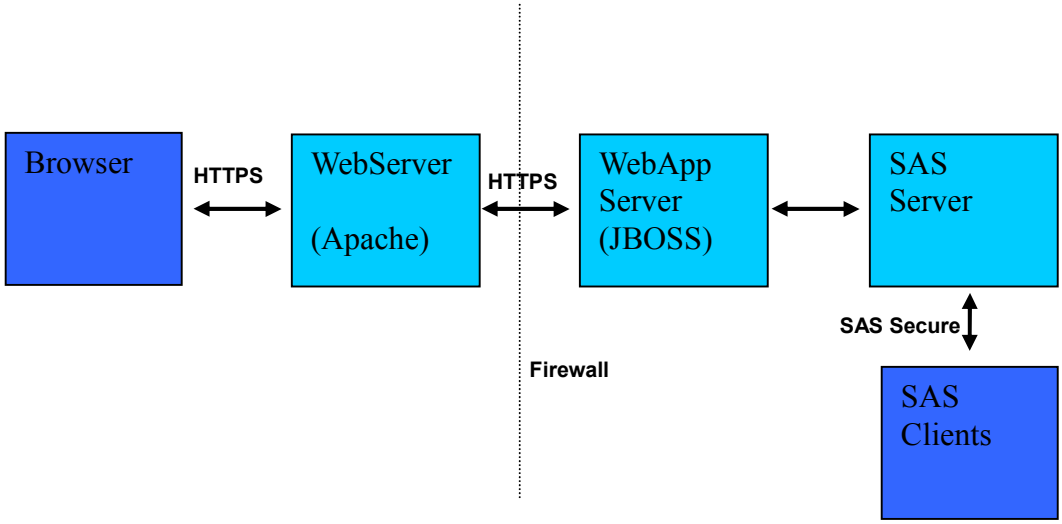
With ever increasing frequency, SAS® software is used in areas where stringent security policies have to be met. One security task is to reduce the need for entering the credentials through the use of Single Sign On. The SAS web applications and the SAS clients are the places where the SAS user is normally forced to enter credentials.

For the purposes of this paper, the security requirements that have to be met are FIPS 140-2. The principals of this requirement and the implications to SAS are presented in a Global Forum Paper [1] from 2012.

For the web components an approach with a reverse proxy will be presented. The base requirements of the TLS/SSL configuration are also explained and also the impact on the configuration to the SAS configuration.

SAS PRINCIPAL ARCHITECTUE IN A SECURE ENVIRONMENT

The principal architecture is based on the FIPS 140-2 requirements. Only the web components are exposed via reverse proxy to the DMZ. The Web Applications are configured for SSO based on the suggestions from SAS Paper 365-2011[2]. The SAS Client Applications use also SSO. The environment that initiates this paper is based on LINUX. For this reason the presented solution is based on LINUX servers.



Picture 1

In this picture, the traffic between the browser and the web server (in this configuration an Apache HTTP Server) uses the HTTPS protocol. In addition, the traffic between the Apache web server and the JBoss web application server is secured with the HTTPS protocol. The connections between the SAS Clients and SAS Servers can be secured with SAS/SECURE. If your site does not have SAS/SECURE licensed, only the SAS Proprietary algorithm is available. This algorithm is only appropriate for preventing the accidental exposure of information[3]. If you have SAS/SECURE available industry standard encryption algorithms such as AES (Advanced Encryption Standard) can be used, and offer much

[1] http://support.sas.com/resources/papers/proceedings12/358-2012.pdf
[2] http://support.sas.com/resources/papers/proceedings11/365-2011.pdf
[3] http://support.sas.com/documentation/cdl/en/secref/65239/PDF/default/secref.pdf

stronger encryption.

To reduce the need to enter the credentials the JBoss web application server can be configured for single sign on. Additionally, SAS clients such as SAS Enterprise Guide® or SAS Data Integration Studio ® can also be configured to use single sign on.

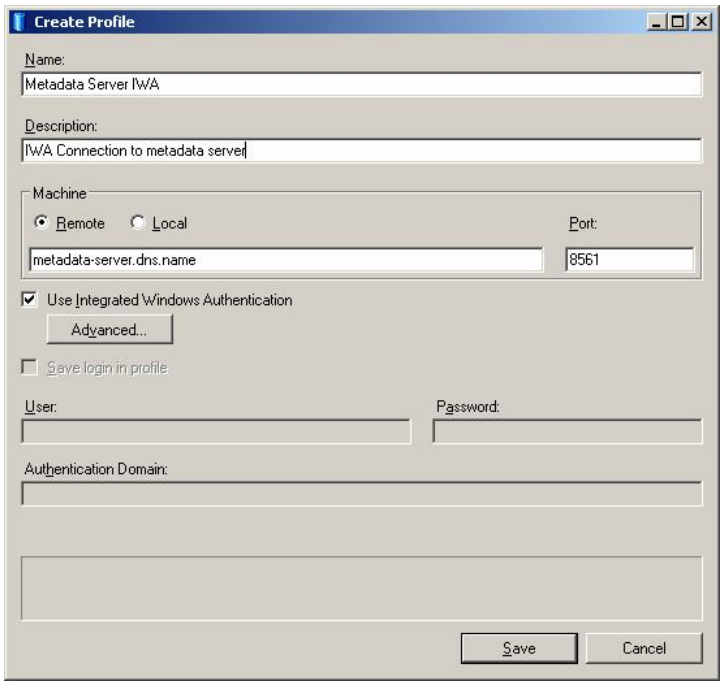
SINGLE SIGN ON CONFIGURATION

For both the SAS client and the SAS web applications an approach with Integrated Windows Authentication (IWA) is used in the presented configuration. Other options for configuring SSO for the Enterprise BI Web applications involve using third party security packages from different vendors, some examples being Tivoli identity Management or CA/Netegrity SiteMinder. The IWA approach uses Microsoft active directory as the authentication provider. Integrated Windows Authentication uses Kerberos as the underlying protocol In this configuration, Active Directory performs the role of the key distribution center (KDC). To configure IWA in SAS in a multi-tier environment, several service principal names (SPN) must be registered in active directory.

1. First the SPN with the URL of the reverse proxy HTTP/xxx.xxx.xx@domain. The created key tab file is used later in the configuration of the JBoss web application server.
2. The other SPN's are required for:
    a. SAS Client products such as SAS Enterprise Guide.
    b. The metadata server SAS/metadata-server-dns.name@domain.
    c. The remaining SPN's are for the SAS compute nodes. It is Important to note that for the SPN, the same DNS name must be used as is configured in the SAS metadata server SAS/compute-1-dns.name@domain and not a different DNS alias.

Explicit details of these configuration steps are available in the following documents: For the SAS Clients: Configuration Guide for SAS ® 9.3 Foundation for UNIX Environments [4](Chapter 5) and for the web access: Configuring Integrated Windows Authentication for JBoss with SAS ® 9.3 Web Applications[5]

Sample: Client Side configuration to enable IWA, select the checkbox (Use Integrated Windows Authentication) in the connection settings.



The other SAS clients such as Data Integration Studio or SAS Management Console have similar checkboxes to enable IWA. Browsers for web access can also be configured for integrated windows authentication. The important thing to remember is that the website URL for the SAS web applications must be configured in the browser as trusted sites. Otherwise the browser still prompts for credentials and the user can't use the single sign on.

[4] http://support.sas.com/documentation/installcenter/en/ikfdtnunxcg/64205/PDF/default/config.pdf
[5] http://support.sas.com/resources/thirdpartysupport/v93/appservers/IWAJBoss.pdf

On the server side there is also some configuration necessary. We have to add the path to the keytab file to the SAS configuration in …/Lev1/level_env_usermods.sh.

Example from the SAS Configuration Guide (or a concrete configuration follow the guidelines in the SAS Documentation[4]):

```
KRB5_KTNAME=/etc/opt/quest/vas/SAS.keytab
export KRB5_KTNAME
```

TLS CONFIGURATION

Transport Layer Security (TLS) is a cryptographic protocol that secures the communication on the application layer. In the proposed architecture approach TLS/SSL is used to secure the communication with the reverse proxy. To provide a convenient approach for the users of the SAS Installation we will use signed certificates. The use of self signed certificates has the disadvantage of end users having to accept an exception to use the SAS Services. For a signed certificate a CA (Certificate Authority) is necessary. The browser needs only the certificates from the CA and not the explicit ones of the reverse proxy. For TLS/SSL a certificate signing request (CSR) is a method to get a signed certificate. This request can be generated with openssl[6]. In addition, the signing of the CSR can be done with openssl. In this case the openssl x509 –req command is used. In this case the access to CA Key is necessary.

Commands to create a CSR: Request and sign this CSR request

Creating a key:

```
openssl genrsa -des3 -out server.key 2048
```

create the CSR: request with the key file

```
openssl req -new -key server.key -out server.csr
```

sign the CSR: request with the CA Keyfile [7]

```
openssl x509 -req –in server.csr –CA certauth.crt –out –CAKey certauth.key –out
server.crt –days 730 –CAcreateserial –CAserial certauth.seq
```

TLS CONFIGURATION FOR THE WEB COMPONENTS

The aforementioned signed certificates will be then integrated into the Apache web server (reverse proxy) configuration. From the architectural approach described in picture 1, we use also SSL from the reverse proxy to the SAS mid-tier. As this communication is internal only (i.e. within the Corporate Network), we can use self-signed certificates for TLS/SSL configuration.
To enable a secure communication between Apache and JBoss, we must enable a redirect to a secure port in the server.xml of JBoss and also the SSL configuration with the path to the certificate files.

On Apache we do not use the mod_jk module described in the SAS documentation[8], as this module does not support end:end SSL. Instead we have chosen the mod_proxy Apache module. For the secure connection to the application server Apache needs the public keys of the self-signed JBoss certificate. In the httpd.conf the ssl for mod_proxy is activated. To ensure the use of SSL a redirect is also configured in Apache.

Redirect permanent / https://reverse-proxy-url/

Every request with http protocol to the Apache is now redirected to the secure https protocol.

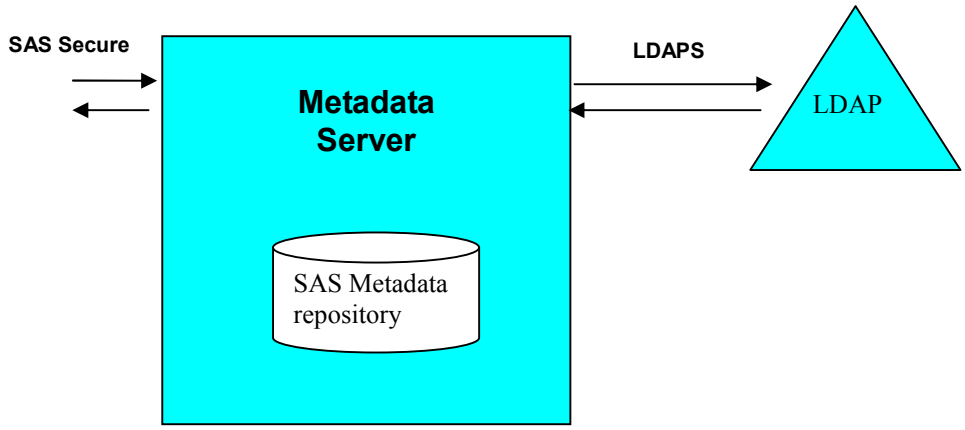
The next step is to change the connections for the SAS Web Applications with the SAS Management console. In the connection tab the host name is changed to the DNS name of the web server and also the port is changed to 443.

TLS CONFIGURATION FOR THE SAS COMPONENTS

The SAS metadata server can be configured to use direct LDAP authentication instead of authentication against the operating system itself. This is an interesting approach, especially if you have components in SAS 9.3 that use the SAS WIP Service (via SAS Web Applications) to connect to the SAS System. One such example is SAS Financial

[6] http://linux.about.com/od/ubusrv_doc/a/ubusg25t12.htm
[7] http://www.herongyang.com/crypto/OpenSSL_Signing_keytool_CSR_4.html
[8] http://support.sas.com/resources/thirdpartysupport/v92m3/appservers/ApacheProxyJBoss.pdf

Management Studio in the SAS Solution SAS® Financial Management. If you configure direct LDAP authentication for the SAS Metadata Server, the users of SAS Financial Management Studio no longer need a user account on the server side.
The first step to configure direct LDAP takes place in the sasv9_usermods.cfg file located in your equivalent of SAS/Config/Lev1/SASMeta/MetadataServer. The explicit changes required are detailed in the paragraph **How to Configure Direct LDAP Authentication** in the Intelligence Platform: Security Administration Guide[9]



To secure the communication from the Metadata Server to the LDAP server a SSL tunnel is used. This type of communication is called LDAPS.

For the SSL configuration of direct LDAP access for the metadata server, add the LDAP_TLSMODE (or AD_TLSMODE) environment variable, and set it to `1`. Setting this variable causes the metadata server to attempt to use SSL. For a trusted SSL connection a Certificate Authority certificate for SSL use is needed. The SAS option to provide this type of certificate is called: SSLCALISTLOC. The option looks like the following example:

```
-SSLCALISTLOC=/…../CA_chain.txt
```

TLS Configuration of Base SAS® Software

From SAS Code several options are possible to access data from a variety of data providers. Some of these connections can also secured by the use of the TLS protocol. In Base SAS there are two types of functions - those that are JAVA based and the others using the standard SAS Kernel.

The SAS JAVA functions use the jproxy tool to launch the JAVA facilities within SAS[10].
In the SAS configuration files (sasv9.cfg) the JRE options are set. In the JRE options a SSL certificate store can be defined. This store enables the SAS JAVA components to trust SSL certificates from other services
Example for a configuration:

```
-JREOPTIONS=(
        -Djavax.net.ssl.trustStore=/opt/sas/sas93/jre1.6.0_21/lib/security/cacerts
        -Djavax.net.ssl.trustStorePassword=XXXX
)
```

To prepare the trust store the keytool[11] utility that comes with the JAVA SDK is used.

The following example shows the commands to import the certificates of the CA into the trust store.

```
cd /opt/sas/sas93/jre1.6.0_21/lib/security/
../../bin/keytool -import -trustcacerts -file /path/to/ca/ca.pem -alias CA_ALIAS -keystore  cacerts
```

To provide the other functions with the trusted certificates the same option is used in the same manner as the connection

[9] http://support.sas.com/documentation/cdl/en/bisecag/63082/PDF/default/bisecag.pdf
[10] http://support.sas.com/documentation/cdl/en/hostunx/63053/HTML/default/viewer.htm#p1rukcdt0nulf5n1bvd9aasodwsf.htm
[11] http://docs.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html

to the LDAP Server detailed earlier in this document.

--SSLCALISTLOC=/…../CA_chain.txt

With this information SAS can open encrypted connection via functions such as the filename statement for https connections.

SECURING THE ACCESS TO DATA SOURCES

To connect to third party databases SAS has the concept of access modules[12] to establish connections between the SAS session and the database. For example to access ORACLE databases SAS uses an ORACLE client on the SAS compute node to establish the connection. To secure connections like these the database client has to be configured for an encrypted connection. One example of Oracle's approach to secure this connection is to use the Oracle advanced security package[13]. In this case the Oracle client on the SAS Compute node is configured to use encrypted network traffic to secure the data transfer. Other database vendors such as IBM with DB2 offer similar concepts to secure the connections to their databases.

CONCLUSION

Using SAS in an environment with enhanced security requirements is a now a more common occurrence because of enhanced data protection regulation. To fully secure a SAS Installation it is important to understand the explicit security requirements of the organization for the desired use of SAS. Based on these requirements the security options to implement can be chosen. In real life architectures, with the consolidation of authentication providers, protection of the user credentials is an important factor. To minimize the need for providing these credentials, this paper recommends the introduction of single sign on. This approach can also be enhanced for the connections to third party databases.
Protection of the most exposed components has the highest priority. Normally these are the web components, and you can secure this connection with SSL. With the use of signed certificates this approach is also very convenient to the end users of SAS.
The securing of SAS system is not a simple project. To achieve a working configuration of SAS 9.3 in such an environment requires a basic understanding of several technologies: First the knowledge on how to handle and create SSL certificates. Second the configuration possibilities of the underlying 3rd party software components like the web application server, web server or the data base clients on the SAS compute nodes. Also the vulnerability testing of system is part of the implementation of secured SAS Installation[14]. This effort allows the use of SAS in use cases with enhanced data protection regulation.

REFERENCES

Security Hardening for SAS® 9.3 Enterprise BI Web Applications
http://support.sas.com/resources/papers/proceedings12/358-2012.pdf

Single Sign-On Configuration and Troubleshooting for SAS® 9.2 Enterprise BI Web Applications
http://support.sas.com/resources/papers/proceedings11/365-2011.pdf

Encryption in SAS® 9.3
http://support.sas.com/documentation/cdl/en/secref/65239/PDF/default/secref.pdf

Configuration Guide for SAS® 9.3 Foundation for UNIX® Environments
http://support.sas.com/documentation/installcenter/en/ikfdtnunxcg/64205/PDF/default/config.pdf

Configuring Integrated Windows Authentication for JBoss with SAS® 9.3 Web Applications
http://support.sas.com/resources/thirdpartysupport/v93/appservers/IWAJBoss.pdf

Ubuntu Documentation
http://linux.about.com/od/ubusrv_doc/a/ubusg25t12.htm

SAS® 9.3 Companion for UNIX Environments
http://support.sas.com/documentation/cdl/en/hostunx/63053/HTML/default/viewer.htm#p1rukcdt0nulf5n1bvd9aasodwsf.htm
ORACLE® JAVA SE Documentation
http://docs.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html

[12] http://www.sas.com/software/data-management/access/index.html
[13] http://www.oracle.com/technetwork/database/options/advanced-security/ds-security-advanced-security-11gr2-1-129479.pdf
[14] https://www.owasp.org/images/5/56/OWASP_Testing_Guide_v3.pdf

ORACE ADVANCED SECUTITY
[1] http://www.oracle.com/technetwork/database/options/advanced-security/ds-security-advanced-security-11gr2-1-129479.pdf

CONTACT INFORMATION

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

Name: Jan Bigalke
Allianz Managed Operations & Services SE
jan.bigalke@allianz.com