Paper 358-2012

Security Hardening for SAS® 9.3 Enterprise BI Web Applications

Heesun Park, SAS Institute Inc., Cary, NC

ABSTRACT

Web configuration for SAS 9.3 Enterprise BI Web applications need to be secured according to an organization's security policy. This paper examines the Web configuration security enhancement options and the protection of Web applications from security vulnerability attacks. Security enhancements for the configuration include single sign-on, integration with a reverse proxy security server, setting up a firewall, the use of SSL, and building a FIPS 140-2 compliant configuration. Implementation of SAS 9.3 Web application protection mechanisms for vulnerability attacks is explained along with the testing process based on the OWASP Top 10 list and the IBM AppScan penetration testing tool.

SINGLE SIGN-ON

SCOPES OF SINGLE SIGN-ON

One simple security enhancement for the SAS 9.3 Enterprise BI (EBI) Web applications is to set up single sign-on (SSO) based on the organization's user authentication infrastructure and policy. Strictly speaking, SSO through SAS Information Delivery Portal® or application server-based SSO is considered SSO, but SSO through third-party security packages like CA SiteMinder or IBM Tivoli Access Manager WebSEAL provides added security for the SAS 9.3 EBI Web applications. Integrated Windows Authentication (IWA) is another form of SSO that uses Windows authentication to access the SAS 9.3 EBI Web applications. By performing user authentication solely through a Windows domain, you make user authentication simple and tight. Good coverage on SSO configuration is available from this SAS Global Forum 2011 paper [1].

The following SAS Technical Support link contains security-related configuration documentation for all of the application servers that SAS supports:
http://support.sas.com/resources/thirdpartysupport/v93/appservers/index.html

After reviewing the basics of protecting Web applications, this paper describes SSO through third-party security packages, and IWA.

WEB APPLICATION PROTECTION MECHANISMS

Securing J2EE Web applications has two aspects. One of these is authentication, which basically controls access to the Web application itself. The other part is authorization, which controls what operations are allowed on resources, such as servers and data, by the authenticated user. The focus of SSO is mainly the authentication process for Web applications, which requires coordination with the existing Web infrastructure. The goal of SSO is to use the authentication mechanism provided by the enterprise through its user registry and to seamlessly integrate this mechanism with SAS® 9.3 Metadata Server user authentication.

J2EE Web applications, including SAS 9.3 Web applications, can be protected in two ways. One way is for the Web application itself to provide its own authentication mechanism. SAS metadata-based authentication (or the SAS host authentication) falls into this category. This method is efficient for a simple configuration that does not require Web perimeter security protection through an external user registry. The other way is to delegate the authentication for the Web application to the application server through the Web application's deployment descriptor. In the deployment descriptor, the authentication method and security role mapping for the Web application can be defined (among other things). This mechanism allows the use of many types of authentication methods as well as integration with the Web space protection security package through an external user registry, such as a Lightweight Directory Access Protocol (LDAP) server. This method supports large and complex enterprise-level security configurations.

Note that the application server can delegate the responsibility for authentication to the Web security package or to the network domain. In other words, the application server "trusts" the other party for user authentication and accepts the authenticated user. Clearly, tight integration between the application server and the authentication provider is required to safely transport the authenticated user information.

The Java Authentication and Authorization Service (JAAS) is the backbone of the authentication process for the Web application server and Web applications. In essence, JAAS is a Java implementation of a Pluggable Authentication Module (PAM). In its simplest form, it is like Java Database Connectivity (JDBC), an abstraction over authentication module providers. To a large extent, JAAS makes Web applications independent from the authentication method and thus makes the Web applications portable. The implementation of JAAS consists of a stack of JAAS login modules that handle different types of authentication methods.

User authentication for Web applications can be done in two ways. One way is for the Web application itself to provide an authentication mechanism with its own JAAS login module. In the case of the SAS 9.3 EBI Web applications, it is called host authentication and is carried out by the OMILoginModule. The other way is to delegate the responsibility for user authentication to other components in the configuration. Initial user authentication can be done by the application server or by a reverse proxy server. In this case, a "trust" relationship is established among the components so that the initial user authentication is honored by the other components. During the process, the application server's login module initializes the JAAS "Subject" object and adds the externally authenticated user into the "Subject" as a JAAS "Principal". The TrustedLoginModule supplied with SAS 9.3 accesses the JAAS "Subject", grabs the authenticated user from the "Principal", and verifies its entry in the SAS Metadata Server. That is the essence of the trusted Web authentication process.

THIRD PARTY SECURITY PACKAGE SSO

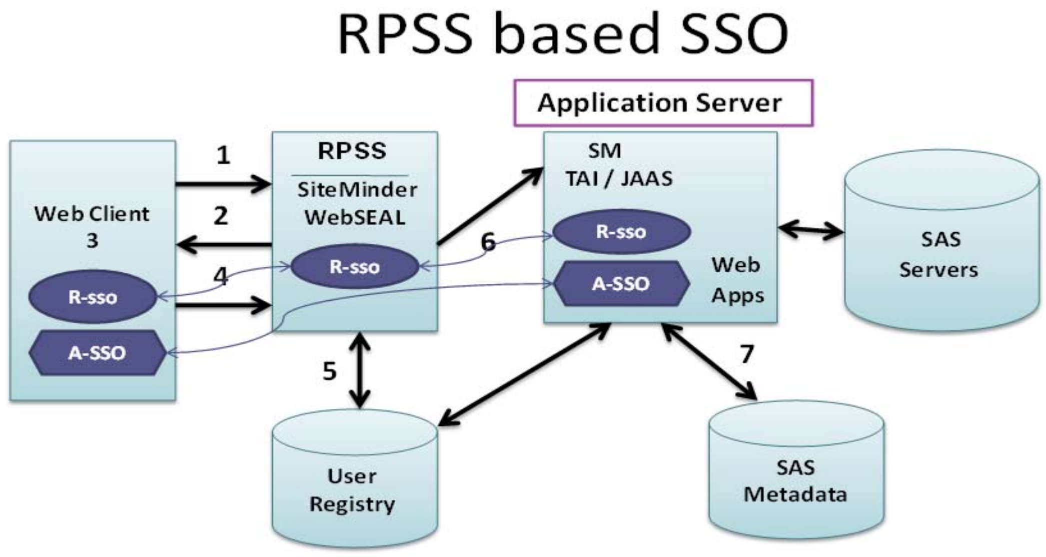
The Web infrastructure of a large organization typically includes a reverse proxy security server (RPSS) that authenticates users before they can access the Web applications deployed to the application server. CA SiteMinder and IBM Tivoli Access Manager (TAM) WebSEAL belong in this category. After the initial user authentication is successful, the RPSS creates a heavily encrypted special token or special HTTP header that contains the user information. This mechanism is considered much safer than the conventional minimally encrypted user name and password method.

To configure a Web application to work with trusted Web authentication through an RPSS, you must first understand how user credentials operate between the RPSS and the application server. Since the application server relies on the RPSS for user authentication, an entity (called an agent or interceptor) that can process the token from the RPSS is needed in the application server. For a WebSphere application server, this entity is called a Trust Association Interceptor (TAI). For an Oracle WebLogic Server, this piece is called an Identity Asserter (IA). In both cases, the user credential does not include a password because the user is already authenticated.

The interceptor module is a special case of a JAAS login module that is placed on the top of the JAAS login module stack and is executed first during JAAS processing. The responsibility of this module is to decode the incoming token from the RPSS, to extract the user name, and to initialize the JAAS Subject with the Principal that represents the authenticated user. The Principal in the JAAS Subject is then consumed by other JAAS login modules including the SAS TrustedLoginModule.

As an authenticator, the RPSS supports SSO from its perspective. The following diagram shows the flow of SSO tokens among the components in the configuration. A special token is created by the RPSS (R-sso token) after the successful user authentication, and it is passed to the browser. The subsequent HTTP requests from the browser carry the token, which is honored by the RPSS. From the application server perspective, after the interceptor module adds the JAAS Principal for the authenticated user, the application server creates its own SSO token (A-sso token). This token also is passed to the browser and allows SSO to the application server from then on.

This process is a lot safer than Basic Authentication since the user information is heavily encrypted in the form of the security token created by the RPSS.

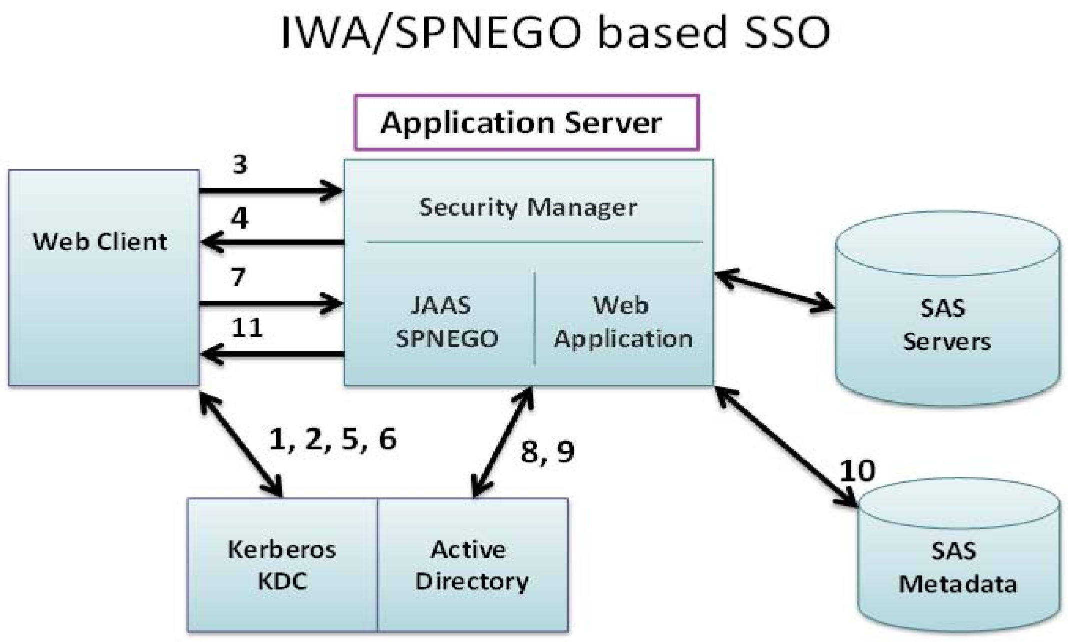


INTEGRATED WINDOWS AUTHENTICATION (IWA) - WINDOWS DOMAIN-BASED SSO

Another popular SSO configuration is the Windows login based approach, which is also called Integrated Windows Authentication (IWA). The underlying authentication protocol is Kerberos. This configuration assumes that clients are on the Windows domain and that the application server can be configured as a Kerberos Service Principal Name (SPN), which means that the application server does not have to be on the Windows domain. In fact, having an application server on UNIX (or Linux) is the most popular choice for IWA configurations.

Good coverage of the Kerberos protocol itself and of IWA configuration for SAS 9.3 EBI Web applications is available from this SAS Global Forum 2010 paper [2].

To comprehend IWA properly, you must understand the underlying protocols, such as the Kerberos and Simple and Protected GSSAPI Negotiation (SPNEGO) protocols. The Kerberos protocol is a ticket-based mutual authentication mechanism developed by the Massachusetts Institute of Technology (MIT) and is the primary authentication protocol for the Windows network (since Windows Server 2003). When a user successfully logs in to the Windows network, the user is represented by his Kerberos ticket. The Kerberos key distribution center (KDC), in conjunction with the domain controllers (DC) in Active Directory, is the centerpiece of Kerberos ticket management. An application server (such as WebSphere) is registered in the KDC with a service principal name (SPN) and becomes a legitimate Kerberos entity. An SPN is mapped to a user and uses his password to decode incoming service tickets. SPNEGO is the protocol supported by the browsers and application servers and is the wrapper around the underlying protocols, such as Kerberos and Microsoft NT LAN Manager (NTLM). Because application servers support only Kerberos, SPNEGO carries only Kerberos tickets. To access a Web application that is deployed in the application server through IWA, a browser requests a Kerberos service ticket for the target application server from the KDC. The data structure of the Kerberos service ticket is very complicated, but in essence, the SPN (application server) uses its password to decode the user information that is embedded in the service ticket, verifies this information, and accepts this information as an authenticated user who can access Web applications. From the application server perspective, this process is a part of its JAAS operation. In the case of WebSphere, it is called SPNEGO Authentication, which is another form of the interceptor approach. This module is executed before any other standard JAAS login module is executed. Other application servers provide a similar approach, but the configuration process is specific to the application server.



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

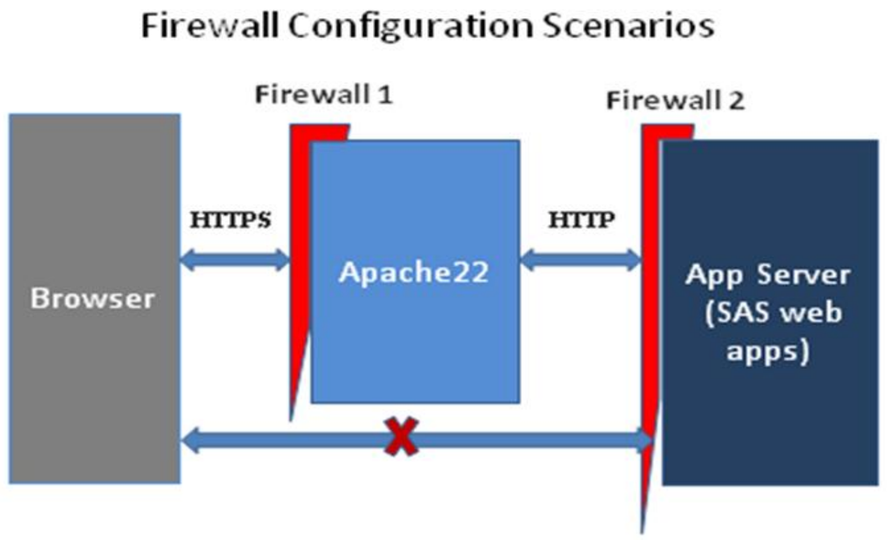
Note that the SPNEGO authentication implementation differs by application server. SAS Technical Support has IWA documentation for every supported application server, which includes necessary modifications to the deployment descriptor for the SAS 9.3 EBI Web application that handles authentication (SASLogonManager).

USE OF FIREWALLS

INTRODUCTION TO FIREWALLS

Firewalls come in many different shapes and forms. An operating system might provide a firewall basically to protect its resources such as the file system. Network switches might provide firewalls inside of them to provide protection from network traffic-related attacks, such as Denial of Service (DoS) attacks. Use of a firewall in Web application configuration protects reverse proxy servers and application servers from unauthorized access from the outside. Using a general purpose firewall such as FireStarter (an open-source firewall), you can limit access to ports and machines by setting up policies and rules. This type of firewall is considered a protocol firewall which is implemented as an IP Router. For example, a firewall can be placed in front of the reverse proxy server (firewall 1) and configured to allow access to port 80 for HTTP (or 443 for HTTPS) only and to block out all other ports. A firewall can also be placed in front of the application server (firewall 2) to protect the traffic in and out the application server. In this case, it can be configured to control access to certain ports such as port 8080 by the reverse proxy server (Apache22) only.

Also note that when the firewall is configured, the preferred transport protocol between the browser and the reverse proxy is usually HTTPS to ensure secure communication to and from sources external to the organization's Web space, whereas the preferred transport protocol between the reverse proxy server and the application server is HTTP to avoid the performance overhead of the HTTPS protocol. This combination of HTTPS and HTTP presents unique situations for generating SAS 9.3 Web application output and handling of the SASTheme_default application. The following sections describe the issues and the solutions.



HTTPS TO HTTP CONVERSION AND REQUESTOVERRIDEFILTER

In this configuration, the traffic to and from Apache22 and the external users (browsers) uses the HTTPS protocol that provides encryption of data to protect its content. But the connection between the reverse proxy server (Apache22) and the application server is based on HTTP. The assumption is that once the traffic reaches the Apache22, it is safely inside the intranet and the chance of an outsider sniffing out the data in transition is very slim, especially with the presence of the firewall. The potential issue for the SAS 9.3 Web application implementation is that some function can call request.getURL(), which will return an internal application server-based URL instead of the originating URL that is anchored on Apache22. The RequestOverrideFilter intercedes in this process and returns the Apache22 location with the HTTPS protocol, which should be used when building the output page. The RequestOverrideFilter is a servlet filter that needs to be added to each Web application that is affected. Since it involves updates to the web.xml file and <web-app>-config.xml file, it is much easier to update the Web application in the “exploded” deployment location. If this is not possible, you must unpack, update, repackage, and redeploy all Web applications.

For more detailed information, contact SAS Technical Support.

EXTERNAL URI FOR SASTHEME_DEFAULT

SAS 9.3 EBI Web applications typically use SASTheme_default during their initialization process. For that reason, we need to set the connection URL of SASTheme_default to that of the application server / port location in the SAS metadata to avoid a potential authentication challenge from the reverse proxy server or third-party security package. But in this secured configuration, the URL of the SASTheme_default in the output page should be based on the Apache22 location. To accommodate this requirement, a new internal (and therefore not well documented) SAS metadata object called External URI has been created under

DeployedComponents/SourceConnection. A new External URI can be added or updated for SASTheme_default in the SAS metadata through a supplied SAS macro or with the SAS Metabrowse utility.

For more detailed information, contact SAS Technical Support.

USE OF SSL AND FIPS 140-2 COMPLIANCE

FIPS 140-2 OVERVIEW

FIPS 140-2 [3] is the documentation published by the National Institute of Standards and Technology (NIST) and is titled "SECURITY REQUIREMENTS FOR CRYPTOGRAPHIC MODULES". The specification contains the guidelines for federal organizations to protect cryptographic-based system configuration for different levels of security. This implies that any cryptographic module used in the system needs to satisfy its security requirements. FIPS 140-2 requires organizations that do business with a government agency or department that requires the exchange of sensitive information to ensure that they meet the FIPS 140-2 security standards. In addition, the financial community increasingly specifies FIPS 140-2 as a procurement requirement.

The security requirements govern the design and implementation of cryptographic modules in the areas of interfaces, authentication, operational environment, cryptographic key management, and mitigation of attacks. Security levels 1, 2, 3, and 4 spell out the use and protection of the cryptographic modules. At a minimum, an approved cryptographic module should be used. Higher security levels require better protection of the cryptographic module itself.

From the Web application and Web infrastructure perspective, FIPS 140-2 compliance means the use of encryption algorithms/modules that are certified to meet the specification. The validation of the cryptographic module is governed by the Cryptographic Module Validation Program (CMVP) that is administered by the NIST and the Communications Security Establishment (CSE) of the Government of Canada.

SSL PROTOCOL AND UNDERLYING TECHNOLOGIES

The foundation of Web application security is the use of SSL with the HTTP protocol, which becomes HTTPS. The use of SSL is probably the most simple and powerful way of achieving secure communication between the Web components. Good coverage of SSL basics and 1-way and 2-way SSL configuration is available from my SSL paper [4].

The SSL handshake is the process of negotiating and selecting a symmetric encryption algorithm to use, and more importantly, creating an encryption key for the encryption algorithm selected. The breakthrough was creating the symmetric encryption key securely and dynamically for each session using Public Key Cryptography (PKC) represented by the X.509 certificate. PKC is an asymmetric encryption algorithm based on number theory where one can encrypt a value (represented by numbers) with one key and decrypt it with the other key. These keys are also known as a public key and a private key. The implementation of PKC uses an X.509 certificate that contains the public key (and a few other things) of the entity it represents. To properly set up and manipulate SSL and certificate-related materials, you must have a solid understanding of the fundamentals of Public Key Cryptography (PKC). From a historical perspective, PKC is the only major revolution in encryption technology in the modern age. The paper [5] published in 1978 by Rivest, Shamir, and Adleman (RSA) at MIT is considered the origin of this technology. The RSA algorithm is the most widely used asymmetric encryption algorithm for PKC. Also, it is important to understand how the X.509 certificate, which is an implementation of the RSA algorithm, is used in the SSL protocol. Note that PKC is used only in the SSL handshake stage, not in subsequent data transmission. The X.509 certificate was designed to ensure the authenticity of the certificate and the integrity of the message during transmission. Symmetric encryption algorithms are available from the cipher suites that the application server and the browser maintain. In essence, the SSL handshake is the process of dynamically and securely selecting the symmetric encryption algorithm and creating the encryption key (session key) for it.

CIPHER SUITE SUPPORT FROM APPLICATION SERVERS

Cipher suites are packaged in the Java Development Kit (JDK). Available cipher suites (encryption algorithms) might differ in each release of JDK. Also provided is the Sun JSSE provider that controls the SSL handshake process and supports the cipher suites in their default preference order. The cipher suite name follows the JSSE cipher suite naming convention, which has the following format: TLS_Kx_[Ka]_WITH_Enc_Bits_MAC, where :

- TLS – SSL Handshake protocol (it should be either TLS or SSL)
- Kx – Key exchange algorithm. RSA and Diffie-Hellman (DH/DHE) are most popular.
- Ka – Key authentication. RSA and DSS are most popular.
- Enc – Symmetric encryption algorithm. Notice that only AES and 3DES_EDE are FIPS 140 approved.
- Bits – Symmetric encryption key size in bits. Less than 128 bits is considered weak.
- MAC – Message Authentication Code. Hash algorithm used for data integrity. SHA is FIPS 140 approved

JBoss and WebLogic application servers typically use a JDK provided by Oracle (Sun). IBM supplies its own JDK for its WebSphere application server, which contains different cipher suites and its own IBM JSSE provider.

Here are some examples of FIPS-approved cipher suites that come with Java SE 6:

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA
- TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA

FIPS 140-2 APPROVED ENCRYPTION ALGORITHMS

According to the NIST Special Publication 800-52 [6], there are a very limited number of encryption algorithms that are FIPS 140-approved. Here is the list:

- SSL handshake protocol: TLS 1.0
- Key Establishment: All of them that are supported by TLS 1.0
- Hash algorithm: SHA-1
- Symmetric encryption algorithm: AES and 3DES-EDE

Clearly, the cipher suites provided by the JDK include a large number of cipher suites that are not FIPS 140-2 approved. Based on the criteria above, it is straightforward to select FIPS 140-2 approved cipher suites. For example, TLS_RSA_WITH_AES_128_CBC_SHA and TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA are FIPS 140-2 approved cipher suites that can be used by the JSSE provider. Since the cipher suites supported for the browser typically are not the same as those supported by the application server, cipher suites on both should be compared, and the ones that are common to both sides should be selected.

FIPS 140-2 COMPLIANCE EFFORT FOR SAS 9.3 SYSTEM

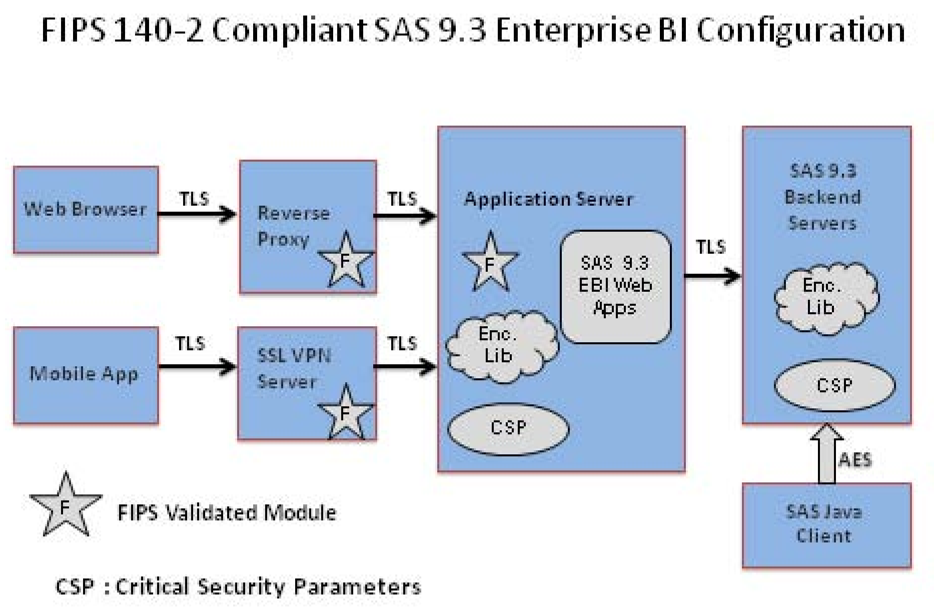
For secure communications among components, the SAS 9.3 system uses encryption in two different ways. One way is to define the specific encryption algorithm to use (FIPS 140-2 approved or not), and the other way is to define SSL as the encryption protocol. Even though it is optimized and most widely used for the HTTP protocol, SSL is application independent and can be used by any application with proper setup. For the FIPS 140-2 compliance for the SAS 9.3, the following distinct levels of security configuration are required:

- Operating systems should have FIPS 140-2 validated modules that use FIPS 140 approved encryption algorithms only. Windows, UNIX, and z/OS support “FIPS mode” operation through system configuration.
- Installation of FIPS 140-2 validated encryption algorithms into the operating system. The OpenSSL project and a number of vendors (including RSA) supply FIPS validated encryption algorithms. SAS has licensed a redistributable RSA encryption library that has been FIPS 140-2 validated.
- Installation of a FIPS 140-2 validated SSL protocol provider into the operating system. Note that this has the same SSL handshake functionality, but it might not be the same SSL provider (JSSE and JCE module) implementation that is supplied by the JDK or by the application server. Refer to the SAS documentation “Encryption in SAS 9.3” [7] for detailed SSL configuration steps for each operating system.

- Setting up of the “encryptfips” option for SAS 9.3 components so that they use FIPS approved encryption for their communication. This is the SAS system-level option. A specific encryption option or SSL option can be set under that “encryptfips” umbrella.

Many SAS products and components have been updated to be FIPS 140-2 compliant. The list includes, but is not limited to, the following: SAS/SECURE™, SAS/CONNECT®, SAS/IntrNet®, metadata, encryption (AES), object spawner and IOM servers, CONNECT spawner, SSL, URL, WebDAV, LDAP connections with SSL, SAS® Management Console , ITConfig, SAS client, SAS® Data Integration Studio, and SAS® Enterprise Guide®.

The following diagram shows the FIPS 140-2 compliance effort for the whole configuration that includes the SAS 9.3 EBI Web applications. It was presented as a case study at ACSAC 2011 conference [8].



SECURITY VULNERABILITY TESTING

WEB APPLICATION VULNERABILITY AND THE OWASP TOP 10

The HTTP protocol was invented long before the Web application. In its original form, it was a stateless protocol and was designed to share and serve static documents. With the advent of session-based Web applications that come with executable JavaScript, a number of HTTP protocol features can be exploited to orchestrate security vulnerability attacks. There have been many efforts to identify and address all potential attacks to make Web applications as safe as possible. For SAS 9.3 Web applications, we are fully aware of the situation and have developed our own protection mechanism and have fully tested our EBI applications with an automated security vulnerability testing tool – AppScan from IBM.

The best organized effort to define these security vulnerabilities is being provided by the Open Web Application Security Project (OWASP), which is non-profit organization. It provides documentation and guidelines on Web application vulnerabilities and how to protect yourself from them. It summarizes its findings in the form of an OWASP Top 10 list. The latest one was published in 2010. Here is the OWASP Top 10 Web Application Security Risks for 2010:
https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

A1: Injection
A2: Cross-Site Scripting (XSS)
A3: Broken Authentication and Session Management

A4: Insecure Direct Object References
A5: Cross-Site Request Forgery (CSRF)
A6: Security Misconfiguration
A7: Insecure Cryptographic Storage
A8: Failure to Restrict URL Access
A9: Insufficient Transport Layer Protection
A10: Unvalidated Redirects and Forwards

Protection from the above security risks requires the coordination of configuration management, the use of security tools such as firewalls and reverse proxy servers, and the protection mechanism in the Web application itself. The most important security risks from the Web application perspective are Injection (A1) and Cross-Site Scripting (A2).

Injection risk is commonly associated with SQL injection in the URL request. When the backend server is based on a database management system and accepts a SQL query, it is possible for attackers to access and steal information off of the database management system with trial and error, blind SQL injection. For SAS 9.3 EBI Web applications, this injection attack is not an issue because our backend server that stores information about SAS resources is our own proprietary SAS Metadata Server with its own API (it is not SQL based). It is not possible to access and steal information from the SAS Metadata Server with blind SQL injection.

Cross-site scripting is the injection of malicious script code in a Web page. The injected code can be executed by the client's browser in a client-side script. The script could run commands to compromise the user's computer. Cross-site scripting is a real threat for the SAS EBI Web applications because any malicious scripting code such as the <script> tag, can be added anywhere in the HTML request page. These malicious tags or codes need to be screened and taken care of before they can be executed.

SANITIZINGREQUESTFILTER FOR SAS EBI WEB APPLICATIONS

To address the cross-site scripting attack for any of our SAS 9.3 EBI Web applications, we implemented a special servlet filter named SanitizingRequestFilter, which is used by all EBI Web applications. A servlet filter is a special mechanism for Web applications that runs before the Web application gets the HTTP request.

The SanitizingRequestFilter works as follows:

- If potentially harmful characters are found on the URL query string, then the request is blocked and an HTTP 403 Forbidden is sent to the client.
- If potentially harmful characters are found in the wrapped request setAttrubute(), getParameter(), getParameterValues(), or getQueryString(), then the value is transformed to safe HTML using the Apache Commons StringEscapeUtils.escapeHtml() method. For example, a > character is translated to &gt;. This prevents it from being translated to hex characters such as %3C that can be used to execute script code in the browser.

It is possible to adjust the filter strength to make it more restrictive. For example, by default, the filter catches the character < or the <script> tag to prevent any inclusion of scripting code. But in some cases, organizations may want to filter the word “script” even though it is not harmful by itself. It is possible to change the SanitizingRequestFilter to catch the word “script” as a potential cross-site scripting risk. In that case, you cannot use word “script” as a valid parameter in any SAS process.

APPSCAN SECURITY VULNERABILITY TESTING

Because there are a tremendous number of variations on the cross-site scripting attack and other security vulnerability testing cases, it is almost impossible to test them manually. SAS has licensed a Web application security vulnerability testing tool, AppScan, from IBM.

AppScan is a security testing tool that can test Web applications and the server infrastructure for security vulnerabilities. The tool tests for the OWASP Top 10 vulnerabilities by generating test cases for known vulnerability attacks and executing them. It has been used for testing SAS 9.2 and SAS 9.3 EBI Web applications. It is being used for SAS Solutions for the same purpose. The AppScan tool's reporting capabilities allow the generation of testing reports based on numerous industry standards, including the OWASP Top 10. We analyze the reports, fix all reported problems, and keep the reports and our assessment for reference to address any specific security requirements or concerns from our customers.

A section titled “Web Application vulnerability testing on SAS 9.3” has been added to the “The Quality Imperative – SAS Institute’s Commitment to Quality” white paper to confirm our security testing effort. This paper can be downloaded from http://www.sas.com/reg/wp/nl/32221.

CONCLUSION

As we examined security enhancement options for SAS 9.3 EBI Web application configurations, we found that there are many choices. It is important to understand the security requirements of the organization first before choosing the right combination of security options to implement. For instance, if user access control is the most important factor, then you have to consider the implementation of SSO or IWA. If the protection of the application server and internal resources is the top priority, then you have to install a firewall to protect them. If you deal with sensitive data and the encryption of data in transport is required, then you have to use SSL (HTTPS) for Web application access. If stronger encryption is needed for the whole configuration, you can follow the FIPS 140-2 guidelines to protect all critical security parameters in the system. Also, any combination of the above security enhancement options is possible.

Typically, securing a system is not trivial. It requires a deep understanding of the underlying technologies as well as the specific implementation of the third-party security packages. SAS Technical Support provides plenty of security-related documentation and advice for the seamless integration and deployment of the SAS 9.3 EBI Web applications in any security infrastructure our customers might have.

REFERENCES

[1] Stuart Rogers and Heesun Park. 2011. "Single Sign-On Configuration and Troubleshooting for SAS 9.2 BI Web Applications." *Proceedings of the SAS Global Forum 2011 Conference*. Cary, NC. SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings11/365-2011.pdf.

[2] Heesun Park. 2010. "Integrated Windows Authentication Support for SAS 9.2 Enterprise BI Web Applications." *Proceedings of the SAS Global Forum 2010 Conference.* Cary, NC. SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings10/312-2010.pdf.

[3] NIST. 2001. FIPS 140-2: *Security Requirements for Cryptographic Modules.* Available at http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf.

[4] Heesun Park and Stan Redford. 2007. "Client Certificate and IP Address based Multi-Factor Authentication for J2EE Web Applications." *Proceedings of the 2007 Conference of the Center for Advanced Studies on Collaborative Research* (CASCON). New York, NY. Available at http://dl.acm.org/citation.cfm?id=1321229.

[5] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,” *Communications of the ACM* 21,2 (1978), 120-126. Available at
http://people.csail.mit.edu/rivest/RivestShamirAdleman-AMethodForObtainingDigitalSignaturesAndPublicKeyCryptosystems.pdf.

[6] NIST. 2005. Special Publication 800-52: *Guidelines for the Selection and Use of Transport Layer Security (TLS) Implementations*. Available at http://csrc.nist.gov/publications/nistpubs/800-52/SP800-52.pdf.

[7] SAS Institute Inc., *Encryption in SAS 9.3*. (Cary, NC: SAS Institute Inc., 2011), 96. Available at http://support.sas.com/documentation/cdl/en/secref/63052/PDF/default/secref.pdf.

[8] Heesun Park. 2011. "Building FIPS 140-2 Compliant Configuration for SAS9.3 BI Web Applications." Annual Computer Security Applications Conference 2011 (ACSAC). Available at http://www.acsac.org/2011/program/case/park.pdf?OPENCONF=377434ec97ef199f06efe94aaa42d107.

CONTACT INFORMATION

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

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