Paper SAS054-2014

Advanced Security Configuration Options for SAS® 9.4 Web Applications and Mobile Devices

Heesun Park, SAS Institute Inc., Cary, NC

ABSTRACT

SAS 9.4 has overhauled web authentication schemes, and the integration with enterprise security infrastructure is quite different from that of SAS 9.3. This paper examines advanced security features such as Secure Sockets Layer (SSL) configuration, single sign-on (SSO) support through Integrated Windows authentication (IWA), and third-party security packages like CA SiteMinder and IBM® Tivoli Access Manager WebSEAL. FIPS 140-2 compliance efforts that enforce the use of a stronger encryption algorithm for web communication and the SAS® system itself are also described. The authentication support for mobile devices such as the iPad is different. The secure Wi-Fi connection from a mobile device to the IT internal resources, as well as how it can be safely integrated into the enterprise security configuration by using the same user repository as the SAS web applications, is explained. The configuration example is shown with SAS® Visual Analytics 6.3.

SAS 9.4 MIDDLE TIER AUTHENTICATION CHANGES

Even though it is not visible to most end users of SAS 9.4 web applications, internal user authentication logic for web applications has been overhauled. SAS Logon Manager that handles the user authentication for all SAS web applications, adopted and customized open-source Central Authentication Service (CAS) [1]. CAS, which is similar to Kerberos, is a ticket-based authentication service. After successful initial user authentication through SAS® Metadata Server or through external web authentication, SAS Logon Manager generates and manages CAS tickets for all SAS web applications with user identity that determines the operation permission level. The CAS ticket and SAS metadata based authorization model replaces traditional security role mapping provided for each web application through deployment descriptor (web.xml file) by the web application server.

Another noticeable change is the inclusion of SAS® Web Application Server (called tcServer, it is the enhanced version of Tomcat 7 by VMware) in the product packages. It comes with many enterprise level management features such as SAS® Environment Manager, and it is free. Since SAS has full control of the web application server, we were able to add more features that make external web authentication support more efficient. Java Authentication and Authorization Service (JAAS) is a separate and independent specification that can be integrated with Java Enterprise Edition (JEE) based web application server.

Use of JAAS-based authentication by web application servers is common, but it requires some investment when you try to integrate with third security packages such as CA SiteMinder or IBM Tivoli Access Manager WebSEAL, because you have to license their JAAS based “application server agent”, which decrypts the incoming authentication token and initializes JAAS “Subject” for consumption by web applications. Similar functionality can be provided by the internal web application server logic called “Valve”. SAS Web Application Server includes SAS developed SiteMinder valve and support of free version of AMTomcatValve from IBM. The valve implementation has less overhead compared to JAAS agent implementation and thus is more efficient.

SECURE SOCKET LAYER BASICS AND CONFIGURATION

SECURE SOCKET LAYER OVERVIEW

When it comes to the protection of data traffic that goes on the unsecured network, use of SSL protocol on top of HTTP (and making it HTTPS) is the first line of defense from any security exposure. SAS 9.4 deployment process includes SSL configuration for some components as options, but it is very important to

understand SSL protocol that is based on public key cryptography (PKC) that is implemented through X.509 certificates [2]. Here are some background of SSL and PKC.

The addition of the SSL protocol to the HTTP protocol has made secure communication on the web possible, and to a large extent made today's e-commerce a reality. The SSL protocol works in two stages. The first stage is the public key cryptography-based handshake during which the two parties that want to communicate agree upon a traditional symmetric encryption algorithm to use in the subsequent communication. The parties also exchange session specific information (a "pre-master secret") in order to generate the encryption key ("session key") that each side will use with the mutually agreed upon encryption algorithm. Because the session key is dynamically generated for each session and is destroyed once the session is terminated, the security exposure of the session key is minimal. The second stage is the use of symmetric encryption with the dynamically generated session key for the rest of the session.

The public key cryptography used in the first stage is an asymmetric encryption scheme that uses a public key and a private key. The public key and its matching private key are calculated based on prime number theory. The content encrypted with the public key can be decrypted only with the matching private key, and vice versa. The delivery mechanism of the public key is the X.509 SSL certificate.

X.509 CERTIFICATE

X.509 is the very widely used SSL certificate standard that defines the fields such as the identity of the owner, the public key of the owner and some other protection mechanisms. There are three types of SSL certificates: server identity certificates, client or personal certificates (we use these terms interchangeably), and certificate authority (CA) certificates. To ensure the authenticity of the certificates, server identity certificates and client certificates must be "signed" by a CA. Signing the certificate means that a one-way hash of the data in the certificate is encrypted with CA's private key. To decrypt and validate the certificate, the consumer of the certificate needs the CA's certificate (that contains its public key) in its "trusted signer" area. These days, most browsers come with well-known CA certificates.

Among other things, an X.509 certificate contains the following fields. The "subject" field is used to identify the owner of the certificate, and the "issuer" field indicates the name of the CA who has signed the certificate. The "Subject Public Key Info" field contains the public key algorithm used and the public key. The "signature" is the message digest (or the hash value of the certificate) encrypted with CA's private key. The "subject" field is used for user identification when client certificate is requested by the server side for authentication (it is called two-way SSL). The creation of SSL certificates has become relatively easy with open-source tools like openSSL [3] from Apache and the Java keytool [4] from Oracle. Small to medium sized organizations can easily create their own CA certificates, server identity certificates, and client certificates for use with their intranet web applications. To accommodate general access from Internet, use of certificates signed by the well-known CA is high recommended.

ONE-WAY SSL, TWO-WAY SSL, AND CLIENT CERTIFICATE AUTHENTICATION

For many security sensitive web applications, such as banking and e-commerce applications, one-way SSL is widely used. In this case, the server sends its certificate to the client (browser) and the client validates server certificate by decrypting the signature (hash value) of the certificate with the public key of the CA that signed the server certificate, agrees on the symmetric encryption algorithm to use, and creates the session key dynamically. Now the secure communication channel is established. Typically, clients are then presented with a logon page where they are required to supply user credentials in order to use the application.

During the SSL handshake, server side can ask for the client certificate, which constitutes two-way SSL. Once the server side receives the client certificate, it has couple of options. One is to validate the client certificate by decrypting the signature with the public key of the CA that signed client certificate and continue with SSL handshake process. The other is to use the client certificate as a means of user authentication. The "subject" field of the client certificate contains user identity in the form of the Lightweight Directory Access Protocol (LDAP) syntax. User identity extracted from the "subject" field can be used for authentication against the user repository of the configuration. This process is called client certificate authentication (CCA). SAS 9.4 does support CCA as external web user authentication method through the Tomcat "valve" implementation. CCA support for SAS 9.4 comes in two flavors based on configuration. When SAS Web Server (enhanced version of Apache) is configured, it needs to set up for two-way SSL and passes on the client certificate to SAS Web Application Server for CCA. For the stand-alone SAS Web Application Server configuration without SAS Web Server in the picture, CCA is triggered by the CLIENT-CERT authentication method defined in <security-constraint> section of the web.xml (deployment descriptor)

file of the SAS Logon Manager application. For more information, refer to the “Enterprise Integration” section of the “SAS® 9.4 Intelligence Platform: Middle-Tier Administration Guide” [5].

SAS 9.4 SINGLE SIGN-ON OPTIONS

For enterprise level configuration, SSO support [6] through organization’s infrastructure is an important factor for the SAS 9.4 web application access. In essence, SSO means that the initial successful authentication by the organization’s infrastructure is trusted and honored by all components in the SAS web configuration such as SAS Web Server, SAS Web Application Server and SAS Metadata server. In SAS 9.4, we do support following SSO options for the web application access through trusted web authentication – IWA, CA SiteMinder and IBM Tivoli Access Manager WebSEAL. General description of each SSO option and how it is implemented in SAS 9.4 SAS Web Application Server is presented below.

INTEGRATED WINDOWS AUTHENTICATION

IWA probably is the most popular single sign-on mechanism for the web application access. The underlying protocol for IWA is the Kerberos protocol [7] that is considered the most secure and reliable mutual authentication mechanism. When you log on to the Windows domain, you basically use ticket-based Kerberos protocol to authenticate yourself to the domain. IWA comes into the picture when you try to access web applications such as SAS Visual Analytics hosted by SAS Web Application Server. The client side should be on the Windows domain but SAS Web Application Server can be on another host, such as Linux. SAS Web Application Server should be defined as a Kerberos entity, called service principle name (SPN). With this configuration, a user logged in to the Windows domain can access SAS Web Application with its own login identity without getting any authentication challenge from SAS Web Application Server and thus called, Integrated Windows Authentication. Here is some background information on the Kerberos protocol. To properly use IWA, we have to understand the underlying protocols, such as the Kerberos protocol and Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) protocols. The Kerberos protocol is the 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 its Kerberos ticket. The Kerberos key distribution center (KDC) in conjunction with the domain controller (DC) that manages Active Directory is the centerpiece of Kerberos ticket management. An application server (such as SAS Web Application Server) is registered to the KDC with a service principal name (SPN) and becomes a legitimate Kerberos entity. An SPN is mapped to a user and uses its password to decode incoming service tickets. SPNEGO is the protocol supported by the browsers and application servers and is the wrapper of the underlying protocols, such as Kerberos or 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 SAS Web 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 (web application server) uses its password to decode the user information that is embedded in the service ticket, verifies, and accepts the information as an authenticated user who can access web applications. From the web application server perspective, this process is a part of its JAAS operation. JAAS login module for Kerberos decodes the Kerberos ticket, fishes out the user information, and passes that user information to SAS web application. Note that IWA is supported natively by the container (SAS Web Application Server) itself and it is not SAS 9.4 specific authentication implementation.

IWA USER DELEGATION FROM THE MIDDLE TIER TO THE SERVER TIER

One new feature added with regard to IWA user authentication in SAS 9.4 is that the IWA user is now delegated from the middle tier to the server tier so that the same user credential can be used to create a SAS server instance, such as SAS workspace server. Previously, a SAS server instance creation required an operating system-based user account that represented the incoming IWA user. The capability of IWA user delegation from middle tier to server tier eliminates the potential resource permission issues related to the fact that one operating system (OS) account could represent multiple IWA users when it comes to SAS server instance creation. One good example is in SAS Visual Analytics 6.3 administration and local data loading. SAS Visual Analytics 6.3 administrator needs to spawn out SAS workspace server to start and stop SAS® LASR™ Analytic Server and should have SAS Visual Analytics administration permission in the SAS metadata. With IWA user delegation from middle tier to server tier, this process is straight forward since there is no OS user mapping required. The same thing is true when a SAS Visual Analytics 6.3 user needs to load the local data into SAS LASR Analytic Server. A user needs to bring up SAS workspace server to load the local data into SAS LASR Analytic Server, use of its own IWA identity is crucial since the user

should be allowed to access the local data that the user has permission to access. The following conceptual diagram shows the history of IWA support with SAS 9 releases over the years.

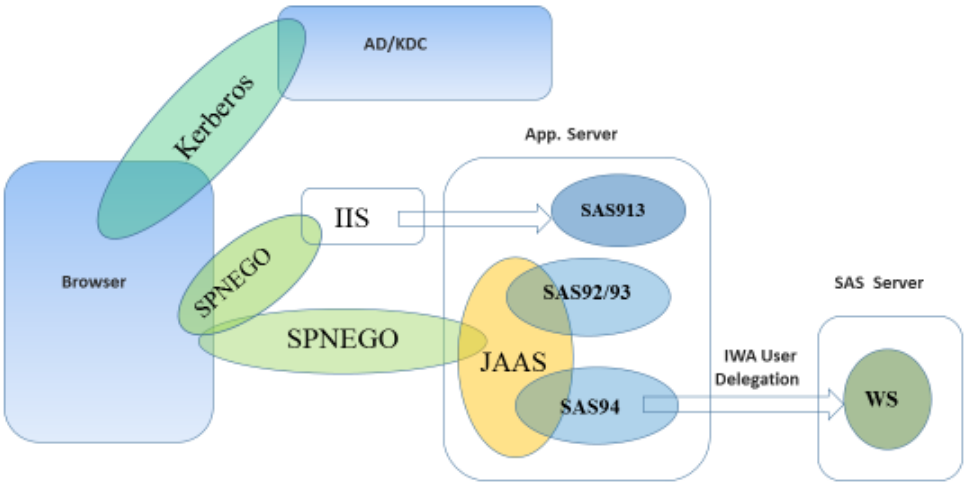

Figure 1. IWA Conceptual Flow Diagram for SAS 9 Releases

THIRD PARTY SECURITY PACKAGE BASED SINGLE SIGN-ON

Many security sensitive organizations protect their web space with web access management (WAM) products such as CA SiteMinder and IBM Tivoli Access Manager WebSEAL. SAS® Enterprise BI Server and SAS Visual Analytics have supported these security packages since SAS 9.2 and SAS 9.3, respectively. SAS continues to support them with the SAS 9.4 based implementation of SAS Enterprise BI Server and SAS Visual Analytics. A major advantage of a WAM product is the use of the single consolidated user repository and the single authentication point for their web space access. Once the user gets successfully authenticated, it creates a session token, such as SMSESSION cookie by SiteMinder or LTPA token by IBM Tivoli Access Manager WebSEAL, which carries the user credentials in heavily encrypted form. It is considered much safer than that of BASE64 encryption used by basic authentication (BA).

Architecturally speaking, the support of these WAM products in SAS 9.4 is quite different from that of SAS 9.2 and SAS 9.3. As it was mentioned earlier, in SAS 9.4, SAS Web Application Server authentication process is not JAAS-based. JAAS, to a large extent, is independent from the container-based authentication. External user authentication for SAS Web Application access in SAS 9.4 uses container provided “valve” (or “adaptor”) implementation to support WAM products. It becomes possible since SAS has a full control of SAS Web Application Server. This implementation is much more cost effective and efficient compared to the JAAS-based approach. For example, in the JAAS-based implementation, our customers have to license the JAAS “agent” module from the WAM provider to plug in to the application server. For SAS Web Application Server, SAS either provides “valve” module or supports the vendor’s module that is needed to decode incoming security token from WAM product.

Here is the authentication sequence:

1. User requests access to protected SAS web application.
2. WAM product intercepts the request and issues an authentication challenge.
3. User provides credentials (user name and password).
4. WAM product receives user credentials.
5. WAM product authenticates the user and creates a security token.
6. Valve in SAS Web Application Server decodes the security token.
7. User is verified against SAS Metadata Server.

The following diagram depicts WAM based SSO in the SAS 9.4 configuration.

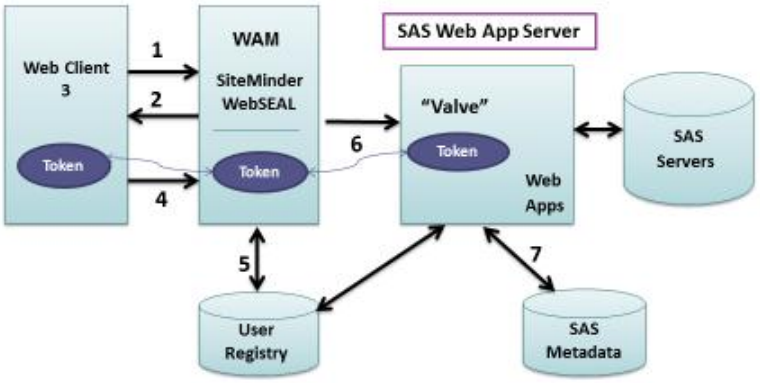

Figure 2. WAM Based SSO in SAS 9.4

FIPS 140-2 COMPLIANCE

FIPS 140-2 [9] is the documentation published by the U.S. National Institute of Technology and Standards (NIST) and is titled “Security Requirements For Cryptographic Modules”. The specification is the guidelines for federal organizations on how 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, approved cryptographic modules should be used. Higher security levels require the 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 and modules that are certified to the specification. The validation of the cryptographic module is governed by the Cryptographic Module Validation Program (CMVP) that is administered by NIST and the Communications Security Establishment Canada (CSE) that is administered by the Canadian government.

The enforcement of FIPS 140-2 applies to many components in web configuration. The focus is on the protection of Critical Security Parameters (CSP) such user name, password, and sensitive data in storage and in transit. Figure 3 shows the locations in the web configuration where FIPS 140-2 approved encryption algorithms should be selected and used. Some components provide FIPS validated modules that force the use of FIPS approved encryption algorithms only, for example, from an SSL handshake. Manual configuration is also feasible to enforce the same behavior. For more comprehensive coverage on this topic, refer to the author’s case study presentation at Annual Computer Security Applications Conference (ACSAC) 2011 titled “Building FIPS 140-2 Compliant Configuration for SAS 9.3 Business Intelligence Web Applications”. [10]

In summary, FIPS 140-2 compliance can be applied in following areas:

1. SSL handshake. Use of Transport Layer Security (TLS) 1.0 protocol for the SSL handshake satisfies the FIPS 140-2 requirement.
2. Reverse Proxy Server (SAS Web Server). Use of FIPS-validated module cand enforce selection FIPS 140-2 approved encryption algorithms during SSL handshake. It can also be configured that way manually.

3. SAS Web Application Server, which is VMware enhanced Apache 2.2.23 uses openSSL module that has been certificated as FIPS 140-2 compliant. Typically, when SAS Web Server is configured to comply with FIPS 140-2, SAS Web Application Server might not need the same type of protection. To use the same type of openSSL module in SAS Web Application Server, the module should be manually compiled and plugged in. Also, SAS Web Application Server can also be configured to use SSL and to support only FIPS 140-2 approved encryption algorithms in its <Connector> definition.
4. On the SAS server-side, communication between the SAS components and protection of CSPs in the system can be achieved by use of SAS global option, -encryptFIPS. It forces the use of FIPS 140-2 approved algorithm (AES) for inter-component communication and encryption of passwords.
5. It is also important to understand that FIPS 140-2 compliance applies not only to the web configuration mentioned here but to the whole computer systems, such as operation system.

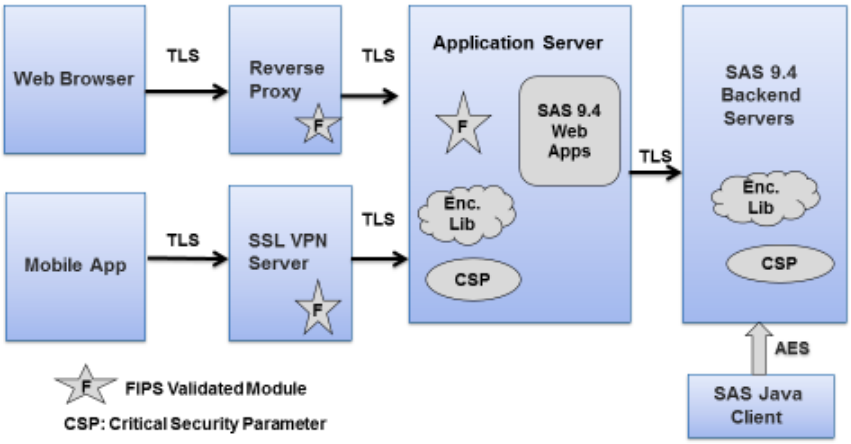

Figure 3. FIPS 140-2 Compliant SAS Web Configuration

PROTECTION FROM WEB APPLICATION SECURITY VULNERABILITIES

The HTTP protocol was invented long before the web application. In its original form, it is a stateless protocol and was designed to share and serve static documents. With advent of session-based web applications that come with executable Java scripts, a number of HTTP protocol features can be exploited to orchestrate security vulnerability attacks. There have been lots of efforts to identify and address all potential attacks to make web applications as safe as possible. For SAS 9.4 web applications, we are fully aware of the situation and we are developing our own protection mechanism and testing our web applications with 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 a non-profit organization. It provides documentation and guidelines on web application vulnerabilities and protections. OWASP also summarizes its findings in the form of the OWASP Top 10 list. The latest list was published in 2013. Here is the OWASP Top 10 web application security risks for 2013: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.

The protection from above security risks requires coordination of configuration management, use of security tools such as firewall and reverse proxy server, and protection mechanism in the web application itself. For SAS 9.3, we have limited AppScan coverage of our web applications such as SAS Enterprise BI Server, SAS Visual Analytics, and a number of solutions. For SAS 9.4 and beyond, we plan to cover all web-based applications with the AppScan security vulnerability tool.

SECURING MOBILE DEVICES AND APPLICATIONS

The surge of mobile devices in the computer industry comes with many challenges in security. We need to understand the security areas of concern that need to be addressed for the mobile devices, and what can be done by whom. Here is the list:

1. System protection. All mobile devices come with basic protection features like passcode lock for the machine and encryption of files in the system. It also provides the capability of remotely cleaning out the contents if the device is lost or stolen. The IT department should be able to control and clean the device that is considered lost or stolen.
2. Encrypted Wi-Fi and virtual private network (VPN) connection. Since a mobile device is a wireless device, wireless traffic for the sensitive data needs to be encrypted. Most Wi-Fi protocols provide many methods of encryption for itself. But from time to time, weakness in the encryption scheme is found and patched. For better protection, many organizations prefer a VPN connection. VPN can be implemented on top of encrypted Wi-Fi protocol as it belongs to the communication layer above the Wi-Fi level. A VPN connection can be shared by multiple mobile applications but the trend is toward a per application-based VPN. The network connection provided by the organization typically enforces the use of encrypted Wi-Fi or VPN.
3. Data encryption by mobile application itself. For example, the SAS® Mobile BI application uses the HTTP protocol for communication to access SAS Web Application Server. Use of the SSL (HTTPS) option enables encryption at the application layer before the data leaves the application. So when all encryption options are used, the traffic in the wireless communication gets encrypted multiple times. It is up to the organization to determine the level of protection it needs for their data.
4. Tethering (SAS data protection). "Tethering" is a SAS Mobile BI application feature for data protection. Instead of the storing actual SAS Visual Analytics reports on the mobile device, it only stores the link to the report on the device. The SAS Mobile BI application fetches the report dynamically from SAS Web Application Server when it is requested after successful login. This feature ensures that if the device is stolen or lost, no one can find any sensitive data on the device.
5. Trusted web authentication support. To access SAS Web Application Server from a mobile device, SAS Mobile BI application requires authentication through SAS Web Application Server. When SAS Web Application Server uses an organization's central user repository such as LDAP server or Active Directory for user authentication, SAS Mobile BI application can be set up to authenticate against the same user repository through its TransportService module in SAS Web Application Server.
6. Support of 3rd party mobile device management (MDM) software. MDM provides fine-grain security provisions per user and per application base. Typically, a mobile application needs to be "wrapped" by the MDM and connects to the MDM server first for user and application authorization before it can connect to SAS Web Application Server. User authentication and authorization by MDM is very much redundant, since it is carried out by the SAS Web Application Server layer and SAS Metadata Server as well. But it is up to the organization to implement extra protection for mobile application access control. SAS supports two MDM packages, one is Good Technology and the other is Mocana.

CONCLUSION

This paper provides an overview of advanced security options that are available to SAS 9.4. New security features include adoption of open-source CAS for better session management, enhanced SSL configuration support through SAS Deployment Wizard, client certificate authentication support with and without SAS Web Server and IWA user delegation from SAS middle tier to SAS server tier. Support of SSO for third party security packages in SAS9.4 uses an application server's internal "valve" mechanism, which is considered more efficient than that of JAAS approach. If the stronger encryption is needed for the whole configuration, you can follow FIPS 140-2 guidelines to protect all critical security parameters in the system. For security vulnerability testing for web applications, SAS plans to expand the AppScan coverage for all web-based applications.

For security of the SAS Mobile BI application, wireless communication encryption, protection of data on the device (tethering), trusted web authentication, and support of third party MDM were explained.

Implementation of security configuration is not trivial. This paper examines most of SAS 9.4 advanced security options. Depending on the organization's security policy and requirements, some or all of the SAS 9.4 advanced security options should be implemented to properly protect SAS and IT resources.

REFERENCES

[1] Central Authentication Service. Available at http://www.jasig.org/cas.

[2] Park, Heesun and Redford, Stan. 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.

[3] OpenSSL – Cryptography and SS/TLS Toolkit. Available at http://www.openssl.org/.

[4] Keytool – Key and Certificate Management Tool. Available at http://docs.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html.

[5] SAS(R) 9.4 Intelligence Platform: Middle-Tier Administration Guide. Available at http://support.sas.com/documentation/cdl/en/bimtag/66823/HTML/default/viewer.htm#p05l4ttthwfpnkn12zd6wshy3wp7.htm.

[6] Park, Heesun and Rogers, Stuart. 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.

[7] Kerberos: The Network Authentication Protocol. Available at http://web.mit.edu/kerberos/.

[8] Park, Heesun. 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.

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

[10] Park, Heesun. 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:

Heesun Park
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
Heesun.Park@sas.com
http://www.sas.com