SAS2540-2016

How to Make Your SAS® Web Applications More Secure: Top Ten Tips

Heesun Park, SAS Institute Inc., Cary, NC

ABSTRACT

Most SAS® products are offered as web-based applications these days. Even though web-based interface provides unmatched accessibility, it comes with known security vulnerabilities. This paper examines the most common exploitation scenarios and suggests the ways to make web applications more secure. The top ten focus areas include protection of user credentials, use of stronger authentication methods, implementation of SSL for all communications between client and server, understanding of attacking mechanism, penetration testing, adoption and integration with third-party security packages, encryption of any sensitive data, security logging and auditing, mobile device access management, and prevention of threats from inside.

INTRODUCTION: OVERVIEW OF SAS 9.4 WEB APPLICATION CONFIGURATION

SAS 9.4 web-based applications are packaged with common SAS 9.4 mid-tier platform applications that include SAS® Web Server and SAS® Web Application Server. It comes with basic user access management through authentication, HTTPS configuration for all major components, and support of data encryption through SAS options and encryption libraries. SAS web application configurations can be integrated with enterprise level security infrastructure, such as identity access management and web access management products. We will look at the security aspect of every component in the configuration and suggest the ways to enhance the security of the whole configuration. The following figure depicts SAS web application configuration integrated into the enterprise level security infrastructure.

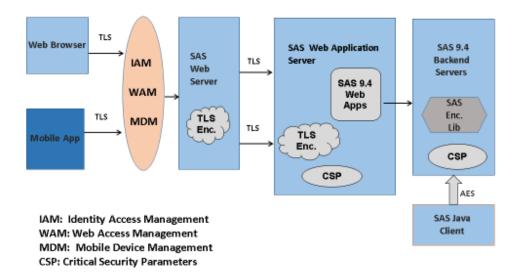


Figure 1 Enterprise Level Security Configuration for SAS Web Applications

PROTECTION OF USER CREDENTIALS

User credentials used in authentication for SAS web application access can be many different forms when we consider the authentication methods support for the SAS configuration. It could be user name/password, client certificate, or specialized tokens generated for the enterprise. User name/password is still the most common authentication method. It is important to select a good password that is difficult to guess. Some of the recommendations include "use a combination of letters, numbers and special characters" and "use the words that cannot be found in any dictionary." The bottom line is that making the user name/password more difficult to guess will mitigate the possibility of brute force hacking attempts. Once we have a good password, we have put some effort to protecting it properly. Best practice would be to memorize it and to not write it down or save it on the computer as is. If you want to save it, you would want to exercise your own simple encryption logic of your own with which you could come up with correct password but no one else can. For example, when your password is Xyz123#, you can save it like X******\$. You should remember that when you have \$, it should be converted to #, when you actually use it. Depending on the number of web applications, machines and configurations that you need to access, you might have to remember several user name/password combinations. If we can afford less user name/password combinations, we could be better off since we have less to worry about. That brings in the idea of Single Sign-On (SSO). In our machine, enterprise infrastructure and web application configuration, there would be multiple layers of authentication points, for example, log in to the machine, authentication through enterprise level security product, web application server, and finally web application itself. SSO means that initial authentication would be honored by the rest of the authentication chains in the configuration. Most SAS 9.4 web applications support SSO through Windows domain based Integrated Windows Authentication (IWA) and/or third-party security package based identity management.

Aside from user name/password, X.509 client certificate can be used as a means of user authentication. Typically, a client certificate is placed in a chip on the identification card. A good example is the Common Access Card (CAC) used by the Department of Defense for military personnel and eligible contractor personnel. The CAC card along with access code can be used as authentication method to access the machine and web application. The CAC card is considered safer than that of user name/password since you have less to remember. In case it gets stolen, it is invalidated by adding it to the Certificate Revocation List (CRL) in the authentication system. Most SAS 9.4 web applications (actually through SAS Web Application Server) support certificate based user authentication.

USE OF STRONGER AUTHENTICATION METHODS

By default, SAS web application configuration comes with SAS Metadata (typically based on OS account) based user authentication using user name and password. This configuration can be converted to use stronger authentication method to make SAS web application access more secure. We will cover three authentication methods that are safer than user name/password combination: Integrated Windows Authentication (IWA), Client Certificate Authentication (CCA), and Multi-Factor Authentication.

INTEGRATED WINDOWS AUTHENTICATION (IWA)

IWA is the most popular Single Sign-On scheme as it uses the Windows login credentials (Kerberos ticket) to access SAS web application. SAS Web Application Server needs to be configured to support IWA integration. 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 SAS Web Application Server does not have to be on the Windows domain. In fact, having an SAS Web Application Server on UNIX (or Linux) is the most popular choice for IWA configurations.

IWA is based on underlying Kerberos protocol, which is the most popular and open-source authentication protocol to date. It is considered safer than user name/password combination because in Kerberos protocol, the password is not transmitted in any form. Simply put, SAS web application access from the browser uses the Kerberos ticket that is generated when you log in to the Windows domain and thus eliminates the authentication challenge from the configuration. You can find a good description of the Kerberos protocol from my 2010 SAS Global Forum paper [1].

To comprehend IWA properly, we 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. The SAS Web Application Server 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). SAS Web Application Server (Tomcat based) supports only Kerberos and SPNEGO carries only Kerberos tickets. To access the SAS 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 SAS Web Application Server perspective, this process is implemented as Tomcat "Valve" operation.

CLIENT CERTIFICATE AUTHENTICATION (CCA)

As we mentioned earlier, X.509 client certificate can be used as a means of user authentication. To better understand CCA, we need to examine how Secure Socket Layer (SSL) protocol works and how X.509 certificates are used in the protocol. Good coverage of SSL basics and 1-way and 2-way SSL configuration is available from my previous SSL paper [2]. 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). 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.

With 2-way SSL configuration, the server side requests client side to send its client certificate. For CCA, the server side can look into the client certificate and extract user information (typically from Common Name in "Subject" field) and use it for user authentication. With this authentication method, users will not be challenged for authentication and no user credentials will be transmitted separately, thus more secure.

MULTI-FACTOR AUTHENTICATION / RISK BASED AUTHENTICATION

When high level of security is warranted for user authentication, there are more options for stronger authentication methods. These options are more of system-level options rather than web application level options. Multi-factor authentication requires something like access token or access code on top of user credentials for user log in to the system or web space. For example, access code can be dynamically created by the third-party security service company, and sent to the user's cell phone as a text message. A user needs to enter that access code as well as user credentials to log in to their system. In most cases, SAS web applications honor system authenticated users (Single Sign-On support) when they are properly represented in the SAS Metadata for SAS resource authorization.

With proliferation of mobile devices, sometimes we need to check more than just user credentials to allow access to our web applications. Risk based authentication leverages more risk factors in the authentication process. This type of service is offered by the Mobile Device Management (MDM)

company. When login attempt is made from a mobile device, MDM can check mobile device identification to see whether the device is properly registered and the geo-location of the device to determine whether the request is coming from authorized location. SAS Mobile BI application can be integrated with most MDM security software.

IMPLEMENTATION OF TLS FOR ALL COMMUNICATIONS BETWEEN CLIENT AND SERVER COMPONENTS

In secure SAS configurations, it is imperative that all data in transit be encrypted not to expose any sensitive information during transmission externally and internally. Any external facing web applications should be configured for HTTPS using TLS protocol. Clearly this is the first line of defense when it comes to web based traffic and is the most effective to date. Be sure to keep up with the latest versions of TLS as some old versions have known security vulnerabilities. If the internal web space is protected properly, use of HTTPS configuration for internal web application access can be optional, but the communication can be encrypted using TLS to protect any sensitive data flowing in clear text form even internally or even on the same machine. TLS handshake protocol is based on the Public Key Cryptography (PKC), which is an asynchronous encryption technology that is implemented through x.509 certificates. The challenges in TLS configuration for web applications and other components come from the fact that the representation, storage and management of x.509 certificate differ by the components in the system. In principle, TLS handshake protocol is the same for all TLS client and server configuration. The difference is how x.509 certificates are managed in each components. For example, in Java-based web applications, .cer format certificates are stored in "keyStore:" while in C-based SAS system, .pem format certificates are stored in flat file. Good coverage of TLS configuration details for various client and server combination is available from my SGF2015 paper [3].

UNDERSTANDING OF ATTACKING MECHANISM AND PENETRATION TESTING

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 many efforts to identify and address all potential attacks to make web applications as safe as possible. One of the organized efforts to define these security vulnerabilities is 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 most common attack mechanism exploits "Injection" vulnerability. Any web-based users whether they are already authenticated or not, can send requests to the target web application with parameters that contain malicious code in them. Injected code can be OS commands, SQL statements, or Java codes. The best approach to block them is the input validation that means that all those potentially malicious codes should be filtered before they can be executed by the web application. The second common attack scenario takes advantage of weak authentication scheme or steals session-related information. For example, if your user name and password are stolen, it is relatively easy for hackers to access your sensitive information. A session cookie is the token that is created after user authentication and represents the user until it gets invalidated. Typically, session cookies are stored in the browser cache to avoid re-authentication of the user each time a request is made. There are number of ways hackers can steal your valid session cookie while you are in a session with SAS web application. For example, if you click on the link from email from the hacker while you are in session with SAS web applications, your browser might send your SAS web application session cookie to a hacker's website. There are some protection mechanisms embedded on the server side, but the best defense is to know the potential danger and protect yourself as much as possible. The third common vulnerability is the Cross Site Scripting (XSS). It exploits the nature of Java Script that allows execution in the browser. There are many

ways that Java Script could be inserted in the HTTP response page. Malicious Java Script can send your session information or other sensitive information from your browser to the hacker's site or their email. Best protection for XSS attack is the use of a filter that screens out any potentially dangerous Java Scripts, even though we might limit some Java Script capabilities.

SAS is fully aware of web application attack scenarios and conducts penetration testing for all web applications before we release the products. As you can imagine, number known vulnerability cases are enormous and it is not feasible to conduct penetration testing in manual fashion. SAS has adopted industry leading penetration testing tools from IBM called AppScan. When we supply all known URLs in the SAS web based product to the AppScan, it examines them and creates test cases for all known vulnerabilities for each and every URL and sends them to the target web application. In most cases, the test cases are in the hundreds of thousands for each product. In some cases, it takes a few days to complete penetration testing for a single product. All reported vulnerabilities are classified and evaluated based OWASP top 10. All critical, high and medium level vulnerabilities are remediated before the product release.

The protection from above security risks requires collaboration from all parties involved such as developer of the web application (SAS), IT infrastructure where the web application is deployed and configured, and the end users that use the applications.

INTEGRATION WITH THIRD-PARTY SECURITY PACKAGES

For enterprise level companies with enterprise wide security infrastructure, it would rather have single web space entry point and authentication-based enterprise user repository than to allow each web application conduct its own user authentication. It means the implementation of Single Sign-On (SSO) for web application access not only for SAS web applications but also other web applications as well. Web Access Management (WAM) products such CA Single Sign-On (formerly known as SiteMinder) and IBM Security Access Manager for Web (formerly known as Tivoli Access Manager / WebSEAL) provide Single Sign-On capability and added protection in the areas of network and user traffic logging and auditing.

Most SAS web based products support integration with above mentioned WAM products. 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. In SAS 9.2 and SAS 9.3, integration with WAM product is based on Java Authentication and Authorization Service (JAAS), which is independent from Web Application Server. As SAS 9.4 includes Tomcat compatible web application server, integration with WAM products gets tightly coupled with container provided "valve" implementation. This implementation is much more cost effective and efficient compared to the JAAS-based approach. It becomes possible since SAS has a full control of SAS Web Application Server. 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.

When integrated with WAM products, SAS 9.4 web application uses this 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.

ENCRYPTION OF ANY SENSITIVE DATA AT REST

The Federal Information Processing Standard Publication 140-2 requires that personally identifiable information be encrypted when it is "at rest." This means that applications must provide support for on disk encryption or use operating system facilities to provide protection for sensitive data. For enterprise level companies with highly sensitive data, they need to leverage options to encrypt that data at rest when it gets stored in storage. There are different levels of encryption for data at rest. For example, encryption of whole machine or whole disk and encryption of whole file system or encryption of specific data in a file or table. In general, operating systems provide capabilities to encrypt data at rest, for example, Windows Encrypting File System (EFS) enables users to encrypt individual files, folders, or entire data drives. Because EFS provides strong encryption through industry-standard algorithms and public key cryptography, encrypted files are confidential even if an attacker bypasses system security. EFS users can share encrypted files with other users on file shares and in web folders. Other operating systems such as Unix/Linux and z/OS provide similar capability.

SAS itself provides encryption of SAS content at rest. The following content can be encrypted at rest: configuration file with passwords, SAS code containing passwords, and SAS data. SAS provides following global security options that can be used with SAS/SECURE.

Option	Description
NETENCRYPT	Force the use of encryption. If a server does not use this, the client can negotiate to not use any encryption.
NETENCRALGORITHM NETENCRALG	Algorithm(s) to use. Specified either as a single algorithm or multiple algorithms in parentheses separated by commas.
NETENCRKEY	Length of encryption key. Only valid for RC2 and RC4 where it can be 40 or 128. If none specified for RC2 and RC4, 128 is assumed.
ENCRYPTFIPS	Perform FIPS 140-2 validation and use FIPS 140-2 validated algorithms only (AES,SSL).

SECURITY LOGGING AND AUDITING

Management of security logging and auditing capability is a very important aspect of overall security implementation of the organization. Logging is crucial to track down unauthorized access to the configuration and to determine if any access attempt was successful or not. Logs also provide individual accountability for the activities. They can be used in reconstruction of events in investigation of program failure or of suspicious activities. It is imperative to log all authentication and authorization-related activities including logging in, logging out, failed login attempts, modification on permission, modification/ deletion of data, and all administrator activities.

SAS components such as SAS Metadata Server and SAS Environment Manager provide various level of security logging option. SAS Environment Manager can collect and produce audit report (log forensics) in many categories:

Access Activity Event Client Context
Access Activity Event Summaries
Access Activity Event Summary by User Account
Access Activity Event Summary without Connection Events
Access activity by user ID, excluding connection activity
Access Activity by User Account
Access Control Changes
Authentication Errors
Group Changes
Logons Not Authorized
Metadata Client Activity
User Accounts Added
User Accounts Removed

If you adopt Web Access Management (WAM) in your configuration, it conducts initial user authentication and authorization. WAM product itself provides high level of security logging capability. For large organizations with heavy traffic to SAS web applications, monitoring and analyzing security logs is not a trivial task. There are number of third-party software based on "Security Information and Event Management" (SIEM) principle, that provides real-time security information analysis and security alerts. Adoption of these tools should be leveraged based on security requirement and policy of the organization.

MOBILE DEVICE ACCESS MANAGEMENT

The proliferation of mobile devices in any organization comes with many challenges in security. We need to understand the security areas of concern when it comes to mobile devices address the issues to conform the security requirement of the organizations. 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 occasionally, 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 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 TLS (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 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 third 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. MDMs can conduct more sophisticated risk-based user authentication by checking the device identity and the geo-location of the device. SAS Web Application Server layer and SAS Metadata Server verify the user identity even after the MDM-based user authentication. SAS supports two MDM packages, one is Good Technology and the other is Mocana.

PREVENTION OF THREATS FROM INSIDE

The human factor is one of the weakest links in computer security. According to industry analysts, about 60% of hacking is related to insider and 80% (of 60%) are unintentional. For example, some administrators of the system use the same username/password as his social networking site. It means that the majority of the hacking could be prevented if we pay more attention to ourselves and to our systems. There is a top 20 "Critical Security Control" (CSC) list documented by Center for Internet Security (CIS). It is worthwhile to check the whole list. Here are a few CSC items that could be used to mitigate threats from inside:

CSC 5: Controlled Use of Administrative Privileges

CSC 6: Maintenance, Monitoring, and Analysis of Audit Logs

CSC 14: Controlled Access Based on the Need to Know

Basically to mitigate insider threats, organizations need to find a way to facilitate detection of suspicious activities by insiders. Level and method of implementation can be controlled by the organization's security policy and practice but clearly more investment in this area would make your system more secure.

CONCLUSION

In this paper, we covered many ways to make our system more secure. Some of them are SAS application specific and others are IT infrastructure components that can be integrated with SAS foundation and application to harden the security of the SAS environment and configuration. To achieve our security goal, it is imperative that we understand the whole security picture in general and make collaborative effort by doing the best possible job based on individual role.

REFERENCES

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

[2] Park, Heesun, 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

[3] Park, Heesun. 2015. "SSL Configuration Best Practices for SAS® Visual Analytics 7.1 Web Applications and SAS® LASR™ Authorization Service." *Proceedings of SAS Global Forum 2015.* Available at http://support.sas.com/resources/papers/proceedings15/SAS1541-2015.pdf

[4] SAS Institute Inc. 2016. SAS® 9.4 Intelligence Platform: Middle-Tier Administration Guide. Available at http://support.sas.com/documentation/cdl/en/bimtag/68217/HTML/default/viewer.htm#p05l4ttthwfpnkn12z d6wshy3wp7.htm

[5] 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

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.