Securing Your SAS® 9 Deployment

Larry Noe, R&D Security Strategist
When it comes to security, we try to build a castle that is impenetrable. We all know how difficult that is to do. There has to be a way to get in and out – probably multiple ways. My goal today is to explain some of the ways you can reduce security exposures. I am working hard every day to make SAS 9 as secure as it can be. And I know we still have work to do in that area – we always will. I know that some of the things I am going to talk about could be done by the configuration wizard at install time. Those items are being worked on to improve the default security of the installation in an “out of the box” security project. Some of the items must be done on site and are specific to the site and IT policies. Those would be very difficult to do in the configuration wizard, but maybe not impossible.

I have a stack of note slides that I am going to make available at the end of the talk, that give more detail than I can discuss up here. I am also going to leave some time at the end for questions, but if something is unclear during the talk, feel free to interrupt and ask a question then. In addition, I will be in the demo area the rest of the day to answer your questions, so please head down to the demo area if I did not cover something you wanted covered or something I said up here is unclear.

I am not going to get into too much detail with metadata permissions, data access permissions or row-level permissions – those would require individual presentations. Again, if you have questions or want to discuss things further, just come find me at the security area on the demo floor. If I can’t answer your question, I am sure I can find someone who can.

I hope this presentation and the notes I am providing will help you make your deployments more secure. You will make your IT people a little happier, and you’ll be able to sleep a little better at night.
We are going to start out with a list of general guidelines and improvements. These are kind of all over the place: some physical security items, some encryption, and some best practices.

Next, we quickly cover best practices in metadata security – and it has to be quick, because we could spend the entire presentation on it.

After that, I will touch on the Unix specific topic of Pluggable Authentication Modules with LDAP/AD.

Lastly, in case those other topics don't keep you riveted to your seats, I will discuss some future directions and improvements to security that are planned for the next major release.
The first thing I want to show you before we get started is the new “SAS 9.1.3 Intelligence Platform Security Administration Guide”. This new Guide is a “must have” and a “must read”. We split the security chapters out of the Planning and Admin Guide, enhanced them, clarified them and added new sections. Please stop by the publications booth and order a copy. I’ve got one in the demo area at the security booth, so come over and take a look at it.
General Guidelines

- Lock down the configuration directory

For Unix and z/OS deployments, the configuration wizard locks the configuration directory structure to the installer user id with only a minimum number of directories opened for general or limited (group) access. For Windows deployments, the default configuration directory is not protected by the configuration program and should be locked down to prevent copying of repository datasets and the potential capture of stored passwords in the repository or server startup files. By default, Windows based servers are installed to run as services under the local SYSTEM account. That would be the only account that requires access to the repository folders as well as a number of other folders in the configuration structure. Chapter 4 of the Security Administration Guide outlines the required directory permissions for best protection to minimize access. If you choose to run the servers under a different account, then that account requires exclusive access to the directory structure and the repository. Sites probably want to allow Administrators access as well. This is a requirement for a secure deployment. The configuration wizard will protect this directory structure on Windows in a future release. Of course, placing the deployment on machines that people do not have physical access to nor interactive login rights is also encouraged.
General Guidelines

- Minimize use of Trusted User account

Trusted user accounts are defined in the trustedUsers.txt file that is read by the metadata server at startup time. The role of this account is to allow middle-tier applications to project already authenticated clients onto the metadata server without forcing those clients to reauthenticate to the metadata server. There are 3 primary uses in SAS 9: it is used for web authentication in our web applications, in the OLAP server for user impersonation, and by Web Report Studio for batch reporting. For example, when a web application is deployed into a secured container (app server), the app server authenticates and authorizes the user to run the web application. The web application is only allowed to run for that user if the user has successfully authenticated. In that case, we allow the user to connect to the metadata server using the trusted user account without reauthenticating. Only a trusted user connection can do this — if the connection was not trusted, the application could not connect on the client’s behalf. So it is a way of projecting client identity onto the back-end without requiring individual back-end credentials. In a similar fashion, the OLAP server authenticates its connecting clients, then impersonates them on the metadata server.

So maybe you can get a feel for how this account could be abused by someone. Having access to it could allow someone to write an application that impersonates any user to the metadata server without their credentials.

So the lesson here is: use the trusted user account only when absolutely required. This account is used for more purposes than originally intended because sites limit the number of service accounts we can create for usage. It is the default account the spawner uses to connect to the metadata server even though the spawner performs no trusted operations (in this release), and it is the default account for the pool administrator (the pool administrator account, though not using trusted operations, does require access to puddle login credentials, so it is fairly privileged). If the site is neither running an OLAP server, not using web authentication, and not doing batch reporting, the trusted user account can be removed from the trustedUsers.txt file. This file is in the metadata server directory of the configuration directory structure. When you do this, it makes the account a normal account with no special privileges. If trusted functionality is required, it is possible to minimize the use of the trusted user account in places where it is strictly required. In particular, another account could be used as the pool administrator, to minimize the possibility of exposure and make for a distinct separation of duties.
This recommendation depends on site policy, but I did want to point out the reasons one may do this beyond a requirement set by IT. The Security Administration Guide explains how to set this up in detail.

SAS/SECURE is a product that packages encryption technology from RSA Data Security. It can be used by all sas 9 client and servers, SAS/Connect and SAS/Share. SAS/Secure supplies a set of industry standard encryption algorithms, currently: RC2, RC4, DES and 3DES. AES is being added in the next major release. My demo laptop here is encrypting all sas traffic with RC4 – so if you stop by the demo room, I can show you how to set it up.

If network security is a concern at your site, or your IT standards require encrypted traffic – which is common for financial institutions – then licensing SAS/Secure is a must. The default algorithm supplied with sas is always available, but is a 32-bit, fixed key, proprietary algorithm. The default installation is set up using this algorithm – since that is the only algorithm we know we have access to – and encryption is enabled for only the initial credential exchange between clients and servers. That is, when a person connects to server, like the metadata server or the stored process server, that credential exchange in encrypted. The problem with this is that all other traffic flows in the clear. If an application makes a request for a login to use for external authentication – for connecting to a database for example – that login will flow over the network in an XML response in the clear. That is where the danger lies – network sniffers could pick these credentials off.

There is a performance penalty when using encryption on connections, but so far, the benchmarks we have run have shown very little performance degradation with the SAS/Secure algorithms– at least on Windows. We know that Windows does some key caching which improves the startup time and key exchange. There may be more impact on Unix and z/OS execution. The sasproprietary algorithm is known to be fairly slow in comparison with the industry standard algorithms, but the key exchange on Unix and z/OS may give sasproprietary an edge on those platforms.
Proc pwencode is a base SAS procedure that converts a plain-text password into an encoded or encrypted form.

The current release uses passwords in configuration files to establish connections to the metadata server on startup of servers and web applications. These passwords are stored in those files in an obfuscated form – merely base64 encoded. The best thing to do, is to replace those encoded passwords with encrypted passwords. This improves the situation somewhat. There is documentation on the support web site as well as the Security Administration Guide (Chapter 5) describing the configuration files that must be modified in the event of a password change. Following this documentation will allow you to modify the default base64 encoding with the encrypted form. On my laptop up here, I have done this, and I can show you what it looks like.

An important take-away though is that no matter how well we encrypt the passwords on disk, the files containing these passwords must be host protected. Any kind of storage of passwords involving a fixed key is vulnerable. It is vulnerable for two reasons: one is that someone could step through the code and find the key, the other is that one could write an application that sends the encrypted password to the server to connect. Our applications are supposed to prevent the use of encrypted passwords from the login prompt, but that is just a UI enforcement. My previous comments about locking down the configuration directory (on Windows, since it is locked down automatically on Unix and z/OS) are especially important. There are more areas that must be protected, and I will talk about them shortly.
General Guidelines

- Minimize the storage of credentials

I could probably spend an entire presentation on this one alone, so I have to be a little careful on how much I say about this. I gave a talk about this at SUGI last year, and I would be glad to discuss details of this with any of you in the demo area. The goal here is to minimize the security exposure and the maintenance burden. The main point is that typically, there is no need to store user passwords in metadata. I realize that there are quite a few caveats, but I’d like to cover the most common case of host authentication in a single domain deployment. There are two reasons why you do not need to store user passwords in that environment: first, the authentication process uses the host OS to check the password prior to the server getting control. After the provider has authenticated the user, only the user id is used to identify the user in the metadata, not the password. Second, most applications cache the incoming credentials for the life of the session and reuse those credentials for authentication to other servers, such as the stored process server, workspace server or OLAP server.

When are you going to need stored passwords? For database access. Though again, some sites can minimize this burden by using group accounts for database access rather than individual accounts. It would be expected that IT would manage the group accounts. Also, for “service” accounts, such as the accounts used to run stored process servers and pooled workspace servers. But those aren’t individual user accounts, those are typically IT maintained as well. For LDAP/AD authentication. But again, sites typically use group accounts to handle that, and they are IT maintained, and not individual accounts. Almost certainly for users when web authentication is being used, but as with LDAP/AD, these are usually group accounts.

Make sure to see chapter 2 of the Security Administration Guide for details on authentication and credential reuse.
General Guidelines

- Minimize host access of service accounts
- Minimize host access of user accounts

What I mean by service accounts are accounts used to run stored process servers, pooled workspace servers, and accounts like sasadm and sastrust. This is just a best practice of giving accounts the least amount of access possible and have them still be functional. On Unix for example, we have no requirement that any of the accounts, service or user, have a login shell (except the installation account, because clearly you must log in to install the software). In addition, some Unixes offer the ability to restrict remote logins, or console logins, or direct logins to the machine. If possible, that should be done. One caveat is that on AIX, we check if the user is allowed to log onto the machine, and if that is disabled, we deny the authentication. This was done after a number of site requests for this ability. Note though that the person does not require remote login capability nor a login shell.

Also note that the spawner does not allow X commands by default. This is important to stop users from running commands on back-end servers. It is especially important to prevent it from stored process servers, which run under an IT identity.

On Windows, any account being host authenticated on the machine running a SAS 9 server will require the "Log on as a batch job" user right. This is because of the Windows API we use to validate the user name and password. User’s do not need interactive logon rights though, so there is no need to grant that for us. It is also possible to minimize exposure by making the service accounts local accounts, rather than domain accounts. Of course, some sites do not allow this, and also, it may not be feasible for puddle logins or stored process server logins to be local accounts if they are expected to do Windows network access in running jobs. Still, local accounts are the safest bet for sasrv, sasadm, sastrust, saswbadm or sasguest. Also, the only account that needs to be an Administrator is the account running the object spawner, and this account is the Local System account by default, so "no" accounts need to be administrators.

Finally, since I am talking about accounts, we have you set up the sasdemo account. The only purpose of this account is to test functionality after an install. It is not used by any application, and nobody relies on it. If you wish, you can enter anything you like in the configuration wizard for this account, and then remove the Person object using the user manager in SAS MC. Don’t bother creating a real account for this user.
If you are deploying SAS9 web applications, you must lock down both the deployment areas in your app server and the areas that are used in building the war or ear files. This is a similar problem to locking down the configuration area. The problem is that web applications contain embedded user names and passwords that are used primarily to bootstrap the applications. The only identity that needs access to the deployed area is whatever identity the app server is running under. In addition, whatever safeguards are available to prevent users from perusing the file system where web apps are deployed should be enabled. This is especially important if web applications are built for web authentication. As I have already said, exposing the trusted user id and password is dangerous. If your web tier is on a separate machine – that is great – lock it down. If it is a shared machine, deny users access to the deployment and build areas.
The default installation sets up one stored process server, with 3 load-balanced connections and a single server login under which to run (by default, sassrv). This is set up as an example deployment, and not necessarily the final way a site may want things. In addition, the configuration wizard sets up a single workspace server that is not pooled. Manual instructions must be followed to create a pooled workspace server, which is strongly recommended for Web Report Studio. These two types of servers behave similarly in that user requests for service run under an IT chosen account, not the individual's credentials. What this means is that all file system access will occur as that IT account regardless of who is running the sas code.

For sites that are using sas datasets, this is an important point to note. It is unlikely that a single stored process server identity will suffice in an enterprise environment. With only a single identity, it would need access to all datasets that any user of the stored process server would need to access. It would be safest to set up multiple stored process servers running under different identities, with different host access capabilities. Datasets could then be host protected to the appropriate identity, which would greatly limit exposure. A similar argument can be made for puddle logins in pooled workspace servers accessing sas datasets.

Additional details:

Database access from a stored process can be accomplished in a couple different ways. The first method is by hard-coding libnames with credentials into the stored process and protecting access to the stored process both physically and with metadata access controls. The second method is to use library pre-assignment. This requires that metadata library access be restricted to the identity associated with the login running the stored process server. For example, the sassrv account is the default account set up to run stored process servers. This login is associated with the group identity of the “SAS General Servers” group. That group would have exclusive access to the metadata defined libraries required for any stored processes run on that server. The same setup would be done for any number of stored process server identities. The third way to accomplish this is to set up the stored process as a parameterized stored process that prompts for RDBMS credentials and uses the resulting macro variables for a libname assignment in the code.

For Windows Integrated Authentication, access would normally take place under the credentials of the stored process server process, not the individual client user. It is possible to register some stored processes with workspace servers instead of stored process servers. This allows the stored process to be launched under the identity of the client and access to SQL Server to be done as the client. The restriction in 9.1 is that a stored process registered with a workspace server cannot produce streaming output.
This ties in very nicely with the previous slide. There are multiple layers of security in any deployment. Metadata security is a layer on top of physical security (and I am including database security in that as well). The assumption is that if physical security is setup to enforce access control, there is nothing that can be done by a user to bypass that access control regardless of the settings in the metadata layer. So, if the metadata says they can have access to a table, but physical access controls prevent such access, they will not get access to the table. If the metadata access controls deny access to a table, but the user has physical access to the table, if that user has the ability to bypass the metadata access controls, they can obtain access to the table. It is vital, given the rich set of RDBMS access tools – sas provided or otherwise – that physical security be in place to prevent unauthorized access.

This is easier to accomplish when access is only permitted through web-based tools and the back-end is on separate machines behind firewalls. This kind of setup allows very limited and controlled access. It is much more of a problem for desktop applications where users can submit arbitrary sas code as with Enterprise Guide or run base sas directly. Database access can be obtained with SQL query tools, bypassing metadata access controls.

I doubt I have to say this, but never forget the physical access controls.
I don’t really have time for an overview of metadata security in this presentation. That would be an entire presentation itself. What I am going to do is review the lock down procedures described in detail in the Security Administration Guide, chapter 5. I would be glad to review more detail of access controls with anyone interested down in the demo room. The goal for most sites is to lock out PUBLIC, or anonymous access, but allow defined users appropriate access to the repository. After running through the instructions generated by the configuration wizard, you have a repository that is wide open for public access with a minimal set of users – just the service accounts – and server definitions. Step one would be to deny public access to the repository. At that point, only the unrestricted user (sasadm) has access to the repository. Next – and this differs a little with the Security Admin Guide – I would load all of my users and groups. We provide sample user loading macros for /etc/passwd files and AD/LDAP directories, but we can really load from any source. You just have to be able to get it into something we can read – which is just about anything. We can talk more about the load and sync code off-line if you like.

The next step is optional, but the recommendation is to set up some small group of individuals as administrators in a group. Then, grant that group broad access to the repository. You would probably also want to place them in the adminUsers.txt file so that they can perform administrative functions on the repository. At this point, you would start to grant privileges to appropriate groups of users. Typically, one would expect a gradual opening of the repository. This has to be done by granting ReadMetadata and almost certainly WriteMetadata in the Repository ACT (also called the Default ACT). I would expect ETL developers and cube builders to need access initially to define servers, libraries, tables and cubes. Followed by information map developers and modelers, building the maps and models to be used by end users. The final step is allowing access to the broader user community. You have two choices on opening up things to a wider audience. You can simply grant permissions in the Repository ACT to the SASUSERS group or grant permissions to individual groups you brought in from your user/group load (or defined yourself). Granting SASUSERS is simpler, because it automatically grants access to all defined users. The other method could require more maintenance because new groups would have to be added to the Repository ACT by hand to grant permissions. Of course, there is better control this way for an administrator. Read chapter 5 carefully for all the details.
One of the things that you must do – and this is discussed in detail in chapter 5 of the Security Administration Guide – is to protect your Access Control Templates and User Groups. The problem is that groups and ACTs obtain their access controls from the repository ACT (also called the “Default ACT”). In general, people require broad WriteMetadata access to the repository in order to create new metadata objects. (I realize that is a problem in and of itself, and that is also being investigated) So people will have the ability to modify ACTs and Groups by default. That is not a good thing. The goal is to allow only certain users or groups that you designate – probably the Administrators group I just talked about – to be able to modify group memberships or ACTs. Typically, this group is the only one that you want to be able to modify these objects. Basically, you deny PUBLIC access and grant Administrators access to lock these objects down. We have a new autocall macro shipping in SP4 called MDUGRPAC (I am fairly certain it is available now from technical support) that can be used to lock down all existing groups. This macro is described in detail in the Security Administration Guide, chapter 7, page 95. Look at the macro and the options it provides. I have run it on my laptop here, and can show you exactly what it does. It basically, creates a new ACT and applies it to all existing groups. The ACT can then be modified as the site wishes, protecting all groups. It has some options, and you can look at the source for more details, or come down to the demo area and I’ll show it to you. The ACT should then be applied to itself, the repository ACT and any other ACTs you have defined.
This is a Unix focused slide. The reason I am including it is that it ties into the area of reduced storage of credentials. In addition, there is a lot of confusion about PAM and LDAP back-ends. We have had plenty of Unix customers tell us that they are using LDAP or AD for authentication, and they want SAS to use it as well. The initial assumption is that they need to set up our LDAP or AD alternate authentication provider. Using one of those alternate providers requires you to maintain some set of Unix user ids and passwords in metadata, which, as I have already mentioned, is a maintenance burden as well as a security exposure. As it turns out, many sites these days, are using LDAP or AD as the authentication mechanism for their Unix systems by using software provided by the Unix vendor, Microsoft or other third party providers like Vintela. Usually, this is accomplished by using Unix’s Pluggable Authentication Module or PAM framework. We have a version of our authentication code that works with PAM – this version is available from technical support.

What this means, is that we do host authentication, by calling the PAM module and the PAM module (or other Unix software) takes care of authenticating the user. We then use standard Unix library calls to obtain the necessary user id and group information. What this gets you is back into a default installation using host authentication, and allows us to fit seamlessly into your existing environment. The end result is a minimum number of stored passwords and a much easier deployment. In addition, users use their standard user id and password instead of having to set up new ones and having IT figure out some way to sync them with their source system.

PAM with LDAP or AD is something I have really been pushing instead of the alternate authentication provider. We have had some problems at some sites with deployments using this software, but it is our goal to make them all work. Most work after just a little effort. There is a paper on PAM integration, written by the Unix host group that I can get you a copy of if you are interested.
Future Directions

- “Out of the box” Security Project
- Single sign-on from Windows desktop
- Improved single sign-on between SAS servers
- Reduced credential management

Quite a few of the items I discussed should be done automatically by the installation process or be defaults in the server. We are working on an “out of the box” security project to cover as many of these items as possible in the next release. For now, you must do more work than we would like to lock down your deployments. We hope to have that fixed as soon as possible.

One of the areas that will help reduce the number of required stored credentials as well as improve the entire user experience, is integration with the Windows single sign-on model using what Microsoft calls, the Security Support Provider Interface or SSPI. Using this API, we will take advantage of Kerberos or NTLM. The basic principal is that once a user has authenticated on their desktop, their identity can be safely passed around in token form between servers in the same or trusted domains. This removes the need for users to log into the metadata server with (typically) the same credentials they just used to log into their desktop. It also reduces the security exposure by avoiding the passing of user credentials over the network, and it supports technologies like smartcards or biometrics, since how the person authenticated to their desktop is not a concern. We still have a fair amount of work to do in this area, and we haven’t yet tackled the problem of middle-tier applications, especially when they are deployed on non-Windows boxes. At a minimum, you should see desktop applications, both Windows and Java, using this single sign-on model to Windows back-end servers.

Another project we are working on for the next release is to improve single sign-on between sas servers. Rather than require unique credential authentication for each of our servers against a configured authentication provider, we will leverage one-time password (OTP) generation technology. This technology will allow a user who has authenticated to the metadata server to authenticate to any sas server by having the application generate OTPs that are validated by the metadata server. This allows seamless connections to the OLAP server, Stored Process Server, Table Server, pooled workspace servers, etc, without stored credentials and without passing system passwords over the network. This can work in conjunction with the SSPI code I just mentioned so that no user passwords are ever stored or transmitted across the network.

All of these technologies taken together will reduce the amount of credential management and security exposure for stored credentials. We are also working on reducing the number of service accounts we require to be set up. We are reexamining our credential use, and are trying to eliminate as many as pre-install checklist accounts as possible. For some deployments, we should be able to get it down to a single account. For more complex deployments, more will probably be needed. Some of that depends on the site’s IT requirements.

I would really appreciate it, if you could stop by the demo floor and let me know what your company’s security requirements/requests are to make sure that we are on track to meet them.