Implementing Smart Card Authentication with Splunk 6.x

Justin Boucher, Senior Sales Engineer, Federal Healthcare

Version	Editor	Notes
1.0	**Justin Boucher**	**Document creation**
1.1	**Eric Popowich**	**- Changes made to apache config. Configuration by Jonathan Fair (SilverRhino LLC) and Ben Koehler (Kinney Group).** **- Added description to note requirement for ip addresses in apache conf, web.conf, server.conf per Justin Boucher**
1.2	**Eric Popowich**	**- Added recommendation to include proxy modules in apache config per Jonathan Fair and Ben Koehler.** **- Turn on outbound httpd connections on SE Linux (Jonathan Fair and Ben Koehler)** **- Modified web.conf.spec to enforce remoteUserMatchExact (Jonathan Fair and Ben Koehler)** **- Added troubleshooting tips (Jonathan Fair and Ben Koehler)**
1.3	**Eric Popowich**	**- Added apache configuration to display DoD banner (content provided by Justin Boucher)**

Abstract:
If you have ever worked in a secure environment, such as the Department of Defense (DoD) you are probably pretty familiar with the issues of integrating and supporting Smart Card or Common Access Card (CAC) authentication. Integrating this type of authentication on any platform has been the cause of many headaches in all IT shops, especially if the application doesn't have native Smart Card authentication support. If you are using Splunk, and require smart card authentication within your environment, you may have seen several answers on http://answers.splunk.com that have addressed this very issue. Although these answers are descriptive, and answer the questions that have been asked by the user, there hasn't been an answer to date that has provided the entire process from start to finish. This paper is meant to address this need, and provide a step by step guide for easily implementing this type of authentication within your environment, whether it is a single instance of Splunk, or a massively distributed environment.

Contents

Making Assumptions

Since there is no way of predicting how every environment is constructed, this paper has written for the mass majority of Splunk users that would run into this type of authentication requirements. That is to say, that this paper will provide these steps as they relate to the DoD, and CAC authentication. However, please note that the same process still applies for all types of Smart Card authentication, but some of the intermediary steps presented may be different for your environment. These should be things that you should already have in place and documented in your environment, and should be easy to replace with processes specific to your environment.

This paper also assumes that you have at least a basic understanding of Public Key Infrastructure (PKI), SSL certificates, Active Directory, and Splunk configuration files. For this process to be completed, you will be instantiating an Apache webserver as an SSL proxy to pass your user's credentials through Splunk. You do not need to be an advanced Apache administrator to do this, but a basic understanding of the concepts listed above will greatly assist you during the implementation process. Notes, and items that may require additional attention, will also be identified to assist you along the way.

NOTE: *It is recommended to read this document in its entirety prior to performing any of these steps, as it will provide insight on items that will be needed or good to know to complete the process as easily and efficiently as possible.*

Process Overview

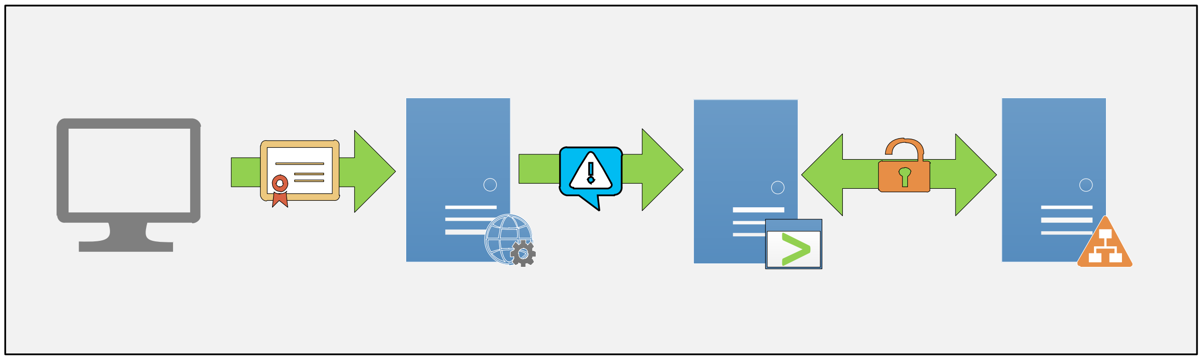

Figure 1 - Smart Card Authentication Process Overview

1. User inserts Smart Card into system and passes client certificate data to the Apache webserver.
2. Apache webserver reads client certificate data and stores user's Unique ID (UID) to a variable.
3. Apache uses a reverse proxy to pass the variable with the UID to Splunk.
4. Splunk reads the UID and sends an authorization request to Active Directory (AD).
5. AD confirms user's credentials and passes access information back to Splunk.
6. Splunk logs in the user with AD information intact.

Testing Environment

Every configuration setting and script within this document has been tested within a secure testing environment. The following applications and security protocols were applied to the testing environment:

Applications

The Splunk implementation used for testing the information within this document contained the following applications:

- Redhat Enterprise 6.5
- Apache 2.2 HTTP Server
  - mod_ssl version 2.4.9
  - mod_proxy version 2.2.4
  - mod_rewrite
- Splunk Enterprise 6.2.2
- OpenSSL 1.0.1e-fips (Heartbleed patch applied)

Security Technical Implementation Guidelines (STIGs)

To ensure that the information would work correctly within a secure environment the following STIGs were applied to the testing server:

- Redhat 6 STIG – Version 1, Release 7
- Apache 2.2 STIG UNIX – Version 1, Release 6

Getting Started

The initial setup process in this section will provide the basic configuration settings required to move forward with implementing the authentication service. If you have a Splunk environment that has been running for a while, you may need to skip some of these configuration steps. As always, please ensure that you properly test this implementation plan in a test environment before applying these changes within production.

The first, and most critical part of this process, is to ensure that your Splunk implementation can access your AD or LDAP environment. For the sake of brevity in this document, the steps to complete this process are not included. You may view the steps to setup LDAP and AD authentication through the GUI using the Splunk Docs link:

http://docs.splunk.com/Documentation/Splunk/6.2.2/Security/ConfigureLDAPwithSplunkWeb

The key step in this LDAP strategy is to use `sAMAccountName` as the **User Attribute Name** and mapping a group with your AD account to the Admin role in Splunk. Once you have completed the steps listed in the document above, you should be able to login to Splunk with a username and password from your AD account. Make sure you can access all the required resources within this account, as you will lose access to the username and password login screen at the end of the Smart Card authentication process. This action will be performed on the Search Head (SH).

NOTE: *If your implementation contains multiple SHs, then you should follow this process on each SH individually. Please refer to the* Authentication Exemptions *section of this document for items that may be exempt from this process.*

SSL Certificates and Certificate Chains

NOTE: *This part of the process will be different for every organization; however, your organization should already have policies in place that define how to conduct the steps that are listed within this section.*

Once you have confirmed that you have access to Splunk through your AD account, and you can access all the data and settings you need to manage your environment, then you can start applying the x.509 certificates to the system. The x.509 certificates allow the Smart Card client certificate to communicate with AD via the server, and provide authorization for accessing your data in Splunk. Since this topic is conducted differently across multiple OS platforms; this document will breakdown the process into two different OS platforms: Windows and Linux/Mac[1].

Defining Root Certificates

In order to apply your own certificate to Splunk's Web Interface (Splunk Web), you must provide a way for the certificate to understand and communicate with your environment. Unless you are using certificates from a commercial Certificate Authority (CA), you will probably need to explicitly provide a certificate chain on the server in which you have installed Splunk. For the DoD this is already provided for you with the DoD Root Certificate package located at: http://dodpki.c3pki.chamb.disa.mil/rootca.html.

Installing the Root Certificates on Windows

Most DoD environments will likely have these certificates packaged within the OS Deployment through SCCM or their standard system imaging process. Please refer to your organization's documentation and Standard Operating Procedures (SOPs) to determine if these certificates are already loaded on your server(s).

If the Root Certificates have not been installed on your system, you can follow the directions provided at http://dodpki.c3pki.chamb.disa.mil/rootca.html.

Installing the Root Certificates on Linux/Mac

Linux and Mac based distributions require a unique process in order to be usable by the system. DISA provides the Root Certificates in p7b format, but Linux and Mac require PEM format in order to group certificate chain information together. This means that each certificate will need to be converted to PEM format, and then added to a master PEM file so that the system understands how to utilize every certificate within the certificate chain. This operation is similar to creating a CRL in Windows. An example script to complete this operation from the download to the conversion has been provided below:

NOTE: *The scripts in this document have been tested and have been confirmed as working in a testing environment. However, you should always test any script, and modify it to fit your environment before using it in production[2].*

[1] Apache and Splunk are native to Linux-based distributions. Every process in this document can be followed on each OS platform, but if your organization allows you to choose the OS to implement Splunk you should investigate implementing this system in its native environment. This will provide the ability to run the SSL Proxy and SH on the same system, with the stability that is provided by running an application in its native environment.

[2] The sample scripts provided here are not supported under any standard support program or service. All scripts are provided AS IS without warranty of any kind and all implied warranties including, without limitation, any implied warranties of merchantability or of fitness for a particular purpose. The entire risk arising out of the use or performance of the sample scripts and documentation remains with you. In no

```
#!/bin/sh

# Obtain certs from DISA and add them to root_certs directory
mkdir root_certs
cd root_certs
wget http://dodpki.c3pki.chamb.disa.mil/rel3_dodroot_2048.p7b
wget http://dodpki.c3pki.chamb.disa.mil/dodeca.p7b
wget http://dodpki.c3pki.chamb.disa.mil/dodeca2.p7b

# Convert .p7b to PEM
for f in *.p7b ; do
  openssl  pkcs7  -inform  DER  -outform  PEM  -in  $f  -out  ${f%.p7b}.pem  -
print_certs
done

# Bundle certs into a single certificate chain
# Add cert to the proper PKI folder
rm -f dod-root-certs.pem
cat rel3_dodroot_2048.pem \
    dodeca.pem dodeca2.pem > dod-root-certs.pem
/bin/cp -p -f *.pem /etc/pki/tls/certs/

```

Script 1 - Convert and Install DoD Root Certificates

Creating and Installing the Splunk Web Certificate

For the DoD Environment, this step will need to be completed based on your organizations policies and procedures. For a Windows based server, you can review the certificate creation steps at the following website: https://technet.microsoft.com/en-us/library/cc732906(v=ws.10).aspx. For Linux and Mac you can review the certificate creation process using OpenSSL by using the following link: https://www.openssl.org/docs/HOWTO/certificates.txt.

In addition, you will need to check your organization's policies to determine if the certificate you applied to your system needs to be registered with your DNS.

Turn on HTTPD Outbound connections

SELinux by default will block httpd outbound connections which is exactly what you want to turn on in order to enable proxying.

/usr/sbin/setsebool -P httpd_can_network_connect 1

event shall, its author, or anyone else involved in the creation, production, or delivery of the scripts be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use of or inability to use the sample scripts or documentation, even if it's author has been advised of the possibility of such damages.

Note that depending on how you do configuration management in your environment this command may not fully persist. (e.g. using ansible/puppet to control your servers)

Installing and Configuring Apache Server in Splunk

Installing Apache

As previously stated in this document, you will be setting up an Apache webserver[3] to create an SSL proxy connection to pass the user's credentials from the Smart Card to the Splunk authentication system. This document will not cover the steps for installing Apache webserver, as the installation steps for Apache are different among all OS platforms. Please refer to your OS specific installation information for installing the base Apache webserver. Ensure that you also install the additional components listed below as they will be required to create the pass-through authentication to Splunk. Some OS specific distributions of Apache may already have these components installed.

Additional Components for Apache

- **mod_ssl:** Creates an interface between Apache and OpenSSL. http://httpd.apache.org/docs/2.2/mod/mod_ssl.html
- **mod_rewrite:** Allows Apache to rewrite the credentials to a variable for Splunk to consume. http://httpd.apache.org/docs/current/mod/mod_rewrite.html
- **mod_proxy:** Creates the reverse proxy interface in Apache. http://httpd.apache.org/docs/current/mod/mod_proxy.html

Configuring Apache SSL

The configuration file for Apache SSL will be the same on any OS platform. The only difference within OS distributions and platforms will be the location of the configuration file on the system. Please refer to your vendor specific information for Apache to determine where to place the SSL configuration file. The SSL configuration created within this document will only contain the SSL configurations specific to the authentication and proxy information. Additional configurations relative to your environment will not be displayed.

Configuration version 1.0

```
<VirtualHost *:443>

ErrorLog logs/ssl_error_log
TransferLog logs/ssl_access_log
LogLevel warn

SSLEngine on

SSLProtocol all +SSLv2 +SSLv3 +TLSv1 +TLSv1.1 +TLSv1.2


SSLCipherSuite
ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+SSLv3:+TLSv1:+EXP:+
```

[3] **WINDOWS USERS:** IIS may also be used as a SSL Proxy engine, but is out of scope for this document. The following link contains resources that may assist in creating the SSL Proxy using IIS version 7.0: https://www.google.com/search?q=iis+7+create+ssl+proxy+for+smart+card+pass+through+authentication.

```
eNULL

SSLCertificateFile /etc/pki/tls/certs/mycertname.mil.crt
SSLCertificateKeyFile /etc/pki/tls/private/mycertname.mil.key

SSLCertificateChainFile /etc/pki/tls/certs/dod-root-certs.pem

SSLCACertificateFile /etc/pki/tls/certs/dod-root-certs.pem

SSLVerifyClient require
SSLVerifyDepth  5

CustomLog logs/splunk_ssl_request_log \
          "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"

ServerAdmin myadminemailaddress@mail.mil
    ServerName 127.0.0.1
    ErrorLog logs/splunk_error_log
    CustomLog logs/splunk_access_log common
    SSLProxyEngine On
    KeepAlive On

    <Proxy *>
        RewriteEngine On
        RewriteCond %{SSL:SSL_CLIENT_S_DN_CN} ([0-9]+$)
        RewriteRule (.*) - [E=USER:%1]
        RequestHeader set cacuser %{USER}e@mil
    </Proxy>

    ProxyPass / https://127.0.0.1:8000/
    ProxyPassReverse / https://127.0.0.1:8000/


</VirtualHost>
```

Configuration 1 – Apache SSL Configuration

Configuration version 1.1 (Fair and Koehler)
These changes have been implemented to remove sslv2 and sslv3. Also updated CipherSuites:

```
SSLProtocol ALL -SSLv2 -SSLv3
SSLHonorCipherOrder On
SSLCipherSuite
ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:ECDH+3DES:DH+3DES:RSA+AESGCM:RSA+AES:RSA+3DES:!aNULL:!MD5:!DSS
```

Configuration version 1.2 (Fair and Koehler)

STIG settings will force you to turn off the proxy modules. When you go back in to turn them on, you will definitely need more than just the base proxy module. I ended up needing to turn on the following two modules:

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
```

Items identified in **BOLD** text will need to be modified to fit your environment. The rest of this section will continue to breakdown the contents of this file by the line number identified to the left of the configuration.

- **Line 1 and 45** – Provides opening and closing statements for creating a Virtual Host in Apache. These will instantiate the communication between the client system and the Apache webserver.
- **Lines 3-5** – Creates a standard log for SSL errors and access attempts to assist in troubleshooting the cause of issues in the configuration.
- **Line 7** – Instructs Apache to use mod_ssl to accept SSL connection requests.
- **Lines 9-12**[4] – Tells Apache which SSL ciphers to accept when a connection is attempted by the client. Refer to the Browser Compatibility Issues section of this document for more information on these settings.
- **Lines 14-19** – Contain the location of the certificates that were created in the previous section, and are referenced as follows:
    - Line 14 – Your certificate name registered from your CA.
    - Line 15 – The key file associated with the certificate from your CA.
    - Line 17 – The file containing your Root Certificate bundle created in the previous section.
    - Line 19 – The CA Root Certificate. If this was bundled within the Certificate Chain, then it would be the same certificate.
- **Lines 24-25** – Creates a custom request log for tracking SSL requests for Splunk. This is not a requirement.
- **Lines 27-32** – Provide your environment specific information, and should be adjusted according to your environment.
- **Lines 34-39** – Contains the configuration to gather the UID information from the client certificate passed to the webserver from the client and place it within a variable in the request header.
    - The ***RewriteCond*** operation will read the *CLIENT_S_DN_CN* information from the client certificate. This client subject information contains all the data about the certificate, to include the user's UID information.
    - ***RewriteRule*** will identify the *USER:* field that is contained within the *CLIENT_S_DN_CN* and store any information presented after the *USER:* field.
    - All the information provided by the *RewriteRule* will be placed inside the header of the request sent to Splunk using the ***RequestHeader*** and set a variable called *cacuser* equal to the *USER* information provided in the previous step. This step will also append the @mil flag to the UID information presented to Splunk to assist Splunk in identifying the proper user.
- **Lines 41-42** – Identifies the location of Splunk to send the *RequestHeader* with the user's UID for login authentication services. Note that you should use IP address and not hostname.

NOTE: *Line 32 contains a KeepAlive statement that allows the session information to be stored. This provides the ability to prevent multiple logins from session timeouts for the end user, but may cause some speed degradation when the data is presented back to the user. It is not required to be enabled, but is more for the end user's convenience. The decision to apply this setting as enabled or disabled is ultimately a question of usability versus performance.*

Browser Compatibility Issues

[4] For the purposes of this document all ciphers have been enabled. Please refer to your organizations policies on which ciphers to enable for SSL.

Splunk supports multiple browsers for accessing searches and data; however, the proxy implementation limits some of this capability. With the DoD restrictions on web browsers and the use of Group Policy Objects (GPOs) to limit the potential for vulnerabilities from browsing the internet, this process limits the compatibility of browser even more. Compatibility information for the three main browsers has been included below:

Internet Explorer
Internet Explorer, or IE, is the most heavily affected by the protocols and procedures within secure environments. The GPOs, browser version restrictions within your organization, and the potential requirement to allow legacy SSL protocols for accessing older web based assets will directly affect how the browser can utilize a proxy to authenticate within Splunk. If you receive an error from Apache after you have completed the entire authentication process within this document; the most likely cause of this issue is the SSLCipherSuite settings within your SSL configuration. Because there is no way to determine the effect the cipher settings will have on your environment, this information cannot be effectively presented in this document. Also note that modifying the SSLCipherSuite settings may affect how other browsers interact with the SSL proxy. For more information about SSL Cipher Suites, click on the following link: http://httpd.apache.org/docs/2.4/mod/mod_ssl.html

Chrome
Throughout all testing of this documentation, there has not been an issue identified within the Chrome family of browsers.

Firefox/Waterfox
No errors have been identified within this browser; however, additional setup is required for reading Smart Cards within this suite of applications. Refer to the following link for step by step instructions on how to configure CAC authentication through these browsers: https://militarycac.com/files/Tech_Note_Firefox_CAC_Authentication.pdf

Configuring Splunk for Smart Card Authentication

The process for configuring Splunk for Smart Card authentication is straight forward. It requires modifying three configuration files with the $SPLUNK_HOME/etc/system directory and restarting the Splunk application. Once these configurations have been correctly modified, you should be able to login to Splunk by accessing the Apache webserver using your server certificate information (i.e. https://mysplunksite.mil instead of mysplunkserver:8000). This section will describe the modifications to each log file individually.

Authentication.conf

The authentication.conf file located at $SPLUNK_HOME/etc/system/authentication.conf contains all of your organization's AD connection information. This was the information you provided to Splunk through the GUI in the Testing Environment

Every configuration setting and script within this document has been tested within a secure testing environment. The following applications and security protocols were applied to the testing environment:

Applications

The Splunk implementation used for testing the information within this document contained the following applications:

- Redhat Enterprise 6.5
- Apache 2.2 HTTP Server
    - mod_ssl version 2.4.9
    - mod_proxy version 2.2.4
    - mod_rewrite
- Splunk Enterprise 6.2.2
- OpenSSL 1.0.1e-fips (Heartbleed patch applied)

Security Technical Implementation Guidelines (STIGs)

To ensure that the information would work correctly within a secure environment the following STIGs were applied to the testing server:

- Redhat 6 STIG – Version 1, Release 7
- Apache 2.2 STIG UNIX – Version 1, Release 6

Getting Started section of this document. There is only a very small, but important, change that needs to be made to the file. In order for Splunk to present the proper authentication data to your AD system, you will need to change the **User Attribute Name** to use the UID provided by the user's Smart Card instead of the sAMAccountName presented in the previous step. Modify your configuration as follows:

```
[Name of your AD Policy]
… SNIPPED …
userNameAttribute = userPrincipalName
… SNIPPED …
```

Configuration 2 - Authentication.conf

The User Principle Name on a DoD CAC contains the UID of the user and the additional @mil required for properly identifying the user in AD. If your organization uses a different field for the Smart Card UID information, you must modify the configuration to present the correct field to AD. If you are unsure with field is used within your environment, you can use ADSI edit to identify the proper field.

Web.conf

Located at $SPLUNK_HOME/etc/system/web.conf, this file provides the entire web based interaction within Splunk. This is the configuration file that the SSL proxy interacts with in order to facilitate the authentication request.

```
[settings]
SSOMode = permissive
remoteUser = cacuser
enableSplunkWebSSL = True
trustedIP = 127.0.0.1
```

Configuration 3 - Web.conf

Breaking this stanza down contains the following information:

- **SSOMode** – Allows strict or permissive access SSO access. See the documentation at http://docs.splunk.com/Documentation/Splunk/6.2.2/admin/Webconf for more information.

- **remoteUser** – This command tells Splunk which user variable to retrieve the user's UID from the request header. The cacuser set in this documentation's web.conf is the same as the variable assigned in the Apache SSL configuration provided above.
- **enableSplunkWebSSL** – This forces the SSL connection for Smart Card authentication to work. This also prevents user access to the default username login screen in Splunk.
- **trustedIP** – The Trusted IP defines which systems can access Splunk Web. You will need to define the IP address of your Apache server, if it is not installed directly on the Splunk server. Note that you should use an ip address and not hostname.

Server.conf

Within this file located at $SPLUNK_HOME/etc/system/server.conf, you will need to identify which IPs are trusted by the system. This should be the same trustedIP information provided within the web.conf file.

```
[general]
… SNIPPED …
trustedIP = 127.0.0.1
… SNIPPED …
```

Configuration 4 - System.conf

Web.conf.spec

Per splunk documentation on web.conf.spec it states that they highly recommend turning on the following setting:

remoteUserMatchExact = 1

This forces it to match exactly the remote username. It had some concerns in the spec file on pre Splunk 6.1 compatibility, which is why it is still off by default. We turned this on as it looks to be a better security practice.

Final Steps

After these configuration files have been modified, you may restart the Splunk service using the `$SPLUNK_HOME/bin/splunk restart` command from your console. After Splunk restarts, you should be able to access Splunk using any certificate from your Smart Card. Note that for trustedIP you should use ip address and not hostname.

Distributed Searching with Smart Card Authentication

The process for using Smart Card Authentication within a Distributed Searching environment is the same as a single instance environment, with a few important caveats. Splunk with distributed searching can work multiple ways within your environment. You may have chosen to logically separate your SHs and split your user base among these SHs, or you could load balance between each SH using a load balancer. Below are some general rules for working with Smart Card authentication and distributed searching:

- If your user base is manually separated and directed to login to a specific SH, you will need to provide multiple certificates. One for each SH within your environment.

- If you are load balancing your SHs, you should off load the SSL proxy operations to your load balancer. If you are using software based load balancers you may be able to install Apache with your load balancer. If you are using a hardware based load balancers like the F5 Big-IP, you can create a custom rule to conduct the SSL Proxy operations.
- If you use an Apache instance that is not directly installed on your Splunk system, you will need to modify the trustedIP settings within the web.conf and system.conf files to contain all IPs from Apache/Load balancers.
- Determine if all SHs require Smart Card Authentication. See Authentication Exemptions for more information.

Authentication Exemptions

There may be times when utilizing Smart Card authentication may not be applicable within your environment. For instance, if you want to use a kiosk system or provide search results on a billboard system or TV, then using Smart Card authentication for these instances would have to be disabled. For these instances you may want to create a separate SH and not configure this SH for Smart Card authentication. This will allow your organization to create an account with username and password access that they can modify the permissions and searching ability to make the system as hardened as possible.

DoD Banner

As far as the consent to monitoring banner is concerned, it can easily be added to something like an F5. Where it gets tricky is trying to insert the banner through apache or another interface. It requires a change to the httpd.conf file for Apache. If they are using CAC login, then they will have to have a Javascript alert that performs the banner operation. Once the user clicks OK, then it would redirect them to Splunk. Here is the code sample for a simple implementation:

```
<VirtualHost *:443>
# Cipher and cert info omitted for brevity

   <Location /cacproxy >
  <Proxy *>
    RewriteEngine On
    RewriteCond %{SSL:SSL_CLIENT_S_DN_CN} ([0-9]+$)
    RewriteRule (.*) - [E=USER:%1]
    RequestHeader set cacuser %{USER}e@mil
  </Proxy>


  ProxyPass https://127.0.0.1:8000/
  ProxyPassReverse https://127.0.0.1:8000/
   </Location>

   # Here is the site with the JS banner
   DocumentRoot /var/www
</VirtualHost>
```

The key change on this is the introduction of the Location tag. It is telling apache to access another folder in the Apache document root to perform the proxy. So the basic workflow is this: The user accesses the URL, and then they are prompted by the JS banner. The JS banner will have a redirect to the /cacproxy page that is identified in the

Location stanza. Once the user is redirected to the https://server_url.mil/cacproxy then the user will be authenticated and redirected to Splunk.

The JS banner page would just be a blank HTML page with the JS code inserted into the head of the document, and would reside in the Apache /var/www directory (according to our example). The banner code for the redirect would be something like this (NOTE: I used a confirmation box here that has a yes/no button, but an alert can be used as well that only has the OK button):

```
if(confirm(³My Consent Banner...")) document.location =
'https://server_url.mil/cacproxy';
```

See, it¹s really easy! LOL. Since I don¹t have ADUC or a CAC to test with, I can only assume that this would work as written. Please have your customer test this out, and we will add it to our Smart Card Authentication white paper. As always, if you have any questions, please let me know.

Troubleshooting Tips

1: https://servername/debug/sso - this is actually documented in the downloadable PDF of the "Securing Splunk" document, but is missing from the website docs. This will help you to quickly see how Splunk is picking up your settings and such.

2: enable dumpio_mod in Apache. This will do *very verbose* logging of your traffic on Apache. This is part of the base Red Hat provided Apache download you just have to turn it on:

```
LoadModule dumpio_module modules/mod_dumpio.so

<VirtualHost *:443>
ErrorLog logs/ssl_error_log
TransferLog logs/ssl_access_log
#Uncomment if full debugging is needed - caution this cause a lot of logging
DumpIOInput On
DumpIOOutput On
DumpIOLogLevel debug
LogLevel debug
#LogLevel warn

........
</VirtualHost>
```

Just shift the commenting to turn it off and set things back to LogLevel warn. The logs for dumpio will show up in the splunk_error_log based on the rest of the configurations already established in the documentation.