DoD Public Key Enablement (PKE) Reference Guide

Public Key Enabling Apache 2.4 HTTP Web Server for Linux

Contact: dodpke@mail.mil
URL: http://iase.disa.mil/pki-pke
URL: http://iase.disa.smil.mil/pki-pke

Enabling PKI Technology for DoD users

Public Key Enabling Apache 2.4 HTTP Web Server for Linux

9 July 2015

Version 2.1

DoD PKE Team

Revision History

Issue Date	Revision	Change Description
1/30/2014	1.0	Initial release
6/12/2014	1.1	Minor content changes
3/19/2015	2.0	General updates and added hint list configuration procedures Appendix E
7/9/2015	2.1	Added steps to enable FIPS mode and additional notes to warn against using insecure versions of OpenSSL

Contents

Introduction

The DoD Public Key Enablement (PKE) reference guides are developed to help an organization augment their security posture through the use of the DoD and National Security Services (NSS) Public Key Infrastructures (PKI). The PKE reference guides contain procedures for enabling products and associated technologies to leverage the security services offered by the DoD and NSS PKIs.

Purpose

The purpose of this reference guide is to provide guidance to the DoD user community on the process to PK-enable an Apache 2.4 web server installed in Federal Information Processing Standards (FIPS) mode for a Linux system. It contains instructions for configuring Apache to use a DoD PKI server certificate, apply appropriate security settings, and implement client certificate-based authentication.

Two modules exist that supply Apache with this functionality: mod_nss and mod_ssl. The DoD PKE team recommends that system integrators use mod_nss instead of mod_ssl for ease of implementation; however, configurations for both modules are presented for completeness. Each section contains information on how to generate a certificate request, submit a certificate request, and retrieve a PKI certificate for the web server. The document contains instructions for unclassified/NIPRNet environments as well as for secret/SIPRNet environments.

It is important that this document is read in its entirety to ensure all relevant steps are performed. Partially implementing the steps in this document can leave portions of a web server unsecured.

Scope

This document is intended for all system administrators. Root privileges are required to complete this guide. No in-depth knowledge of PKI is required. Experience installing and configuring software on Linux platforms is required.

Baseline

This guide was developed using Apache 2.4.4 installed on a CentOS 6.2 Linux system. Users are not bound to the CentOS 6.2 distribution of Linux as the implementation steps in this guide are general for all Apache 2.4 configurations on Linux systems. It is highly recommended that the latest version of Apache be used and the Linux operating system have the latest software updates. The system must also have the latest Apache web server and UNIX Security Technical Implementation Guide (STIG) settings applied. The STIGs are located at http://iase.disa.mil/stigs/.

The command utilities used in this guide are specific to CentOS, Red Hat, and Fedora systems. If this guide is being implemented on different operating systems, equivalent commands will need to be used to accomplish each specific task in the guide.

The text editor used in this guide is "vi". If a different text editor is used simply perform the equivalent operation with the alternate text editor.

Getting Started

NOTE: In this guide, the '$' indicates a user input command in a terminal window and '●' indicates output from the command. '\' at the end of a line of command indicates the next line belongs to the same command and should be entered in the terminal window immediately following the preceding line without a line break.

Verify Apache HTTP service is currently installed on the system. The procedures in this guide are specific to Apache version 2.4. Verify the Apache version is at least 2.4.

If httpd is not installed or an older version is installed, the correct version will need to be installed.

NOTE: If you are using the older version of Apache (2.2), then please refer to the *Public Key Enabling Apache v2.2 HTTP Web Server for Linux* guide which is available on the DoD PKE Engineering website at http://iase.disa.mil/pki-pke under PKE A-Z > Guides.

At the time this guide was created, the "yum" repository did not have version 2.4 of Apache HTTP server available for download. Confirm what version is available in the "yum" repository with the following command:

```
yum list httpd
```

If the output indicates an older version of Apache is installed, Apache 2.4 will need to be built from source. Apache 2.4 can be downloaded at http://httpd.apache.org/download.cgi#apache24 (copy and paste URL to browser).

NOTE: If mod_ssl will be leveraged, it is important to note that Apache 2.4 comes packaged with mod_ssl built in. In order to use the built-in mod_ssl module, the mod_ssl flags need to be specified during the compilation of the source. Be sure to specify `-enable-ssl` when running the configure script.

OpenSSL Vulnerabilities WARNING!

Security vulnerabilities have been discovered in older versions of OpenSSL. Please ensure the latest and most secure version of OpenSSL is used. Refer to www.opensssl.org for the latest updates.

mod_nss vs mod_ssl

Public key enablement of an Apache HTTPD web server can be accomplished through integration of either mod_nss or mod_ssl. mod_nss was created by Red Hat so the Apache web server could use the same security libraries as the former Netscape server products acquired by Red Hat [1]. mod_ssl was derived from the open source [2] SSL/TLS toolkit OpenSSL [3]. Both mod_ssl and mod_nss can be configured to operate in FIPS mode. mod_nss is FIPS compliant and no additional steps are required. In order to enable mod_ssl in FIPS mode, the OpenSSL FIPS Object Module must be compiled to integrate with OpenSSL and the HTTPD service.

For the mod_nss procedures refer to **Configure Apache HTTP with mod_nss**.

For the mod_ssl procedures refer to **Configure Apache HTTP with mod_ssl**.

Configure Apache HTTP with mod_nss

The following procedures describe the steps to public key enable an Apache HTTP web server [4] with mod_nss in FIPS mode [5].

FIPS Mode

The NSS certificate database can be placed in FIPS compliant mode using the modutil command. This will be executed in the section **Create Private Key and Certificate Signing Request**.

Installing mod_nss

The mod_nss module will need to be installed as a plug-in to the Apache HTTP server. If the Apache server was built from source then mod_nss will need to be built from source. When configuring the mod_nss build the Apache build will be specified using the `--with-apxs` flag. This flag will point to the specific Apache instance which mod_nss should plug into. The mod_nss module can be found here at https://fedorahosted.org/released/mod_nss/. See the mod_nss documentation for build guidance.

NOTE: mod_nss has a dependency on the NSS Cryptographic module. Some versions of mod_nss require a specific version of the NSS Cryptographic module. Ensure that the version of mod_nss being used is a version that is compatible with a NSS Cyptographic module that is FIPS Validated according to NIST, see here: http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140val-all.htm .

NOTE: Using 'yum' to install mod_nss will not work if Apache was built from source. The 'yum' installation will install mod_nss into the system Apache (if available) which is most likely different from the Apache which was built from source.

Trusted Certificate Configuration

The following steps describe the process for obtaining and installing the trusted DoD or NSS certificates into the Apache mod_nss database.

NOTE: The default install directory for the base system used in this guide is /usr/local/apache2; in the event that another directory is being used, you will need to modify these instructions to reflect your alternate install directory.

1) Ensure the httpd service is off. Turn it off with the following command:

```
$ /usr/local/apache2/bin/apachectl stop
```

2) Locate the NSS security database directory by referring to the nss.conf file. Use the following command:

```
$ grep NSSCertificateDatabase /usr/local/apache2/conf/nss.conf
```

- `NSSCertificateDatabase /usr/local/apache2/conf`

3) Create a new directory for the NSS security database.

   ```
   $ mkdir /usr/local/apache2/alias
   ```

4) Change the location of the NSS security database in the nss.conf using a text editor.

   ```
   vi /usr/local/apache2/conf/nss.conf
   ```

   Change the default NSSCertificateDatabase by changing to following line (the default may vary):

   ```
   NSSCertificateDatabase /usr/local/apache2/conf
   ```

   to

   ```
   NSSCertificateDatabase /usr/local/apache2/alias
   ```

5) Create the new NSS security database. Use the following command:

   ```
   $ certutil -N -d /usr/local/apache2/alias
   ```

   When prompted, set/confirm a password for key3.db (your private key database) that meets the following NIST FIPS 140-2 requirements [6].

   - The password must be at least seven characters long.
   - The password must consist of characters from three or more character classes. We define five character classes: Digits (0-9), ASCII lowercase letters, ASCII uppercase letters, ASCII non-alphanumeric characters (such as space and punctuation marks), and non-ASCII characters. If an ASCII uppercase letter is the first character of the password, the uppercase letter is not counted toward its character class. Similarly, if a digit is the last character of the password, the digit is not counted toward its character class.

6) Confirm the user that will be used for the Apache service by referring to the `http.conf` file (located in `/usr/local/apache2/conf`). In the file, look for the **User/Group** section. The system user specified for the **User** flag is the user the httpd service will use. Make note of this user for the next step.

7) Ensure the security databases are set with the correct permission. Use the following commands (replace **apache** with the httpd user):

   ```
   $ chown root:apache /usr/local/apache2/alias/
   $ chown root:apache /usr/local/apache2/alias/*.db
   $ chmod 660 /usr/local/apache2/alias/*.db
   ```

8) Obtain the PKI CA certificates for installation into the NSS server database.

   <u>Unclassified/NIPRNet Systems:</u>

The NIPRNet DoD PKI CA certificates can be obtained as PKCS#7 files from the DoD PKE Engineering web site at http://iase.disa.mil/pki-pke under **Tools > Trust Store**. Select the appropriate certificate bundle under **PKI CA Certificate Bundles: PKCS#7.** Save the package locally. Extract the zip file and navigate into the extracted directory structure.

Follow the instructions in the README.txt to validate the signature of the PKCS#7 package.

Secret/SIPRNet Systems:

The SIPRNet/NSS PKI CA certificates can be obtained as PKCS#7 files from the SIPR DoD PKE Engineering web site at http://iase.disa.smil.mil/pki-pke under **Tools > Trust Store**. Select the appropriate certificate bundle under **PKI CA Certificate Bundles: PKCS#7**. Save the package locally. Extract the zip file and navigate into the extracted directory structure.

Follow the instructions in the README.txt to validate the signature of the PKCS#7 package.

9) Install the certificates into the NSS database using the .p7b file and the NSSdb CertLoader tool:

NOTE: The .p7b and .p7c files are the same in the Linux context. It does not matter which extension is applied.

Unclassified/NIPRNet Systems:

Navigate to http://iase.disa.mil/pki-pke/ and download the **NSSdb CertLoader** script under **Tools > Trust Store Management**. Save the script in the folder containing the .p7b files. Refer to the **NSSdb CertLoader User Manual** for instructions on installing and using the **CertLoader** tool. The user manual is also available from the DoD PKE website at http://iase.disa.mil/pki-pke under **Tools > Trust Store Management**.

Run the script from within the folder containing the .p7b file by typing the following command (file name will vary):

```
$ bash NSSdb_CertLoader_Linux.sh Certificates_PKCS7_v4.0.1_DoD.pem-signed.p7b
```

Secret/SIPRNet Systems:

Navigate to http://iase.disa.smil.mil/pki-pke and download the **NSSdb CertLoader** under **Tools > Trust Store Management**. Save the script in the folder containing the .p7b files. Refer to the **NSSdb CertLoader User Manual** for instructions on installing and using the **CertLoader** tool. The user manual is also available from the DoD PKE website at http://iase.disa.smil.mil/pki-pke under **Tools > Trust Store Management**.

Run the script from within the folder containing the .p7b file by typing the following command (file name will vary):

```
$ bash NSSdb_CertLoader_Linux.sh Certificates_PKCS7_v4.0.1_NSS.pem-signed.p7b
```

NOTE: The script will save the .cer certificate files to the present working directory. To list the CA certificates in NSS, run the following command. Initially there should not be any certificates in the database.

```
$ certutil -L -d /usr/local/apache2/alias | sort
```

Create Private Key and Certificate Signing Request

The Apache HTTP server will require a DoD or NSS PKI server certificate to be PK-enabled. The following steps detail how to obtain the server certificate.

1) Place the NSS certificate database in FIPS mode:

```
$ modutil -fips true -dbdir /etc/httpd/alias
```

2) Generate the server key pair and certificate signing request using the following command (navigate to a location you would like the request to be stored):

Note: The <site-name> below should be the fully qualified domain name of the server that external clients would use to access it; for example, iase.disa.mil. The <CC/S/A> should be completed as appropriate for your organization; for example, DISA.

NIPRNet:

```
$ certutil -R -s "cn=<site-name>,ou=<CC/S/A>,ou=PKI,ou=DoD,o=U.S. \
Government,c=US" -o req.p10 -g 2048 -d /usr/local/apache2/alias/ -a
```

SIPRNet:

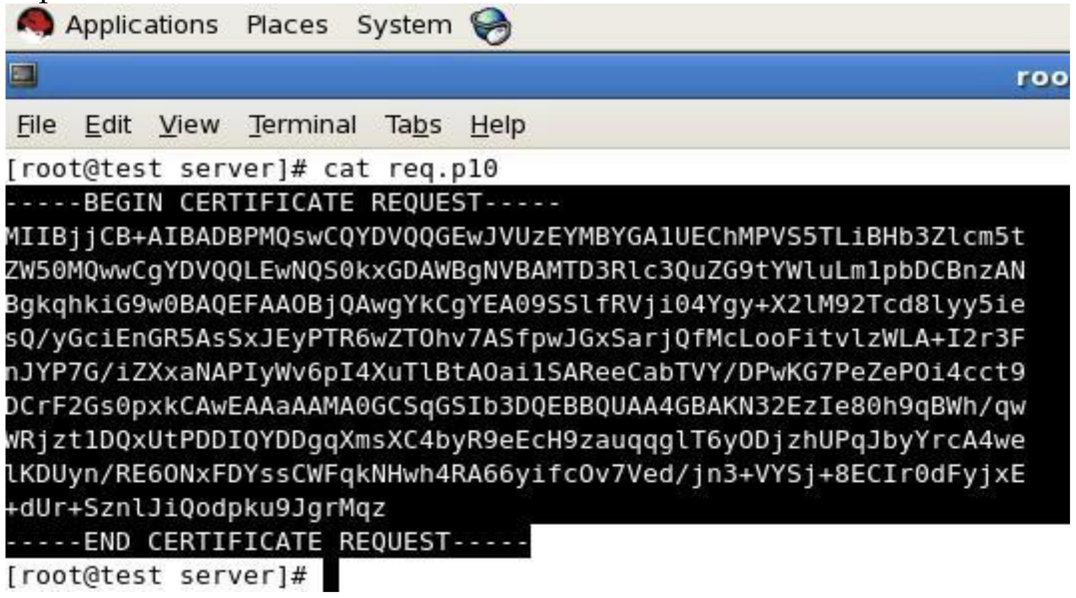
```
$ certutil -R -s "cn=<site-name>,ou=<CC/S/A>,ou=DoD,ou=NSS,o=U.S. \
Government,c=US" -o req.p10 -g 2048 -d /usr/local/apache2/alias/ -a
```

The output prompt will prompt to type randomly until the progress bar has completed. Press Enter when it is complete.

3) Display the Base 64 encoded certificate request so that it may be copied and pasted into the certificate request submission form. Use the command:

```
$ cat req.p10
```

Copy the request including the “-----BEGIN CERTIFICATE REQUEST-----“and “-----END CERTIFICATE REQUEST-----“. This will be pasted into the certificate request submission form on the CA website.



```
[root@test server]# cat req.p10
-----BEGIN CERTIFICATE REQUEST-----
MIIBjjCB+AIBADBPMQswCQYDVQQGEwJVUzEYMBYGA1UEChMPVS5TLiBHb3Zlcm5t
ZW50MQwwCgYDVQQLEwNQS0kxGDAWBgNVBAMTD3Rlc3QuZG9tYWluLm1pbDCBnzAN
BgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA09SSlfRVji04Ygy+X2lM92Tcd8lyy5ie
sQ/yGciEnGR5AsSxJEyPTR6wZTOhv7ASfpwJGxSarjQfMcLooFitvlzWLA+I2r3F
nJYP7G/iZXxaNAPIyWv6pI4XuTlBtAOai1SAReeCabTVY/DPwKG7PeZePOi4cct9
DCrF2Gs0pxkCAwEAAaAAMA0GCSqGSIb3DQEBBQUAA4GBAKN32EzIe80h9qBWh/qw
WRjzt1DQxUtPDDIQYDDgqXmsXC4byR9eEcH9zauqqglT6yODjzhUPqJbyYrcA4we
lKDUyn/RE6ONxFDYssCWFqkNHwh4RA66yifcOv7Ved/jn3+VYSj+8ECIr0dFyjxE
+dUr+SznlJiQodpku9JgrMqz
-----END CERTIFICATE REQUEST-----
[root@test server]#
```

4) The next step is to submit the request. For NIPRNet, instructions on how to submit a SSL server certificate request can be obtained from the DoD PKE web site at http://iase.disa.mil/pki-pke. For SIPRNet, instructions on how to submit a SSL server certificate request can be obtained from the SIPRNet PKE Web site at http://iase.disa.smil.mil/pki-pke . Navigate to **For Administrators, Integrators & Developers > Web Servers** and select the guide **Obtaining a PKI Certificate for a DoD Server**. Use this guide to submit and retrieve the request generated above. Name your downloaded certificate `server.cer`.

Server Certificate Installation

Once the DoD PKI certificate is obtained, use the following command to install the certificate into the NSS database:

```
$ certutil -A -n Server-Cert -i server.cer -t "u,u,u" -d /usr/local/apache2/alias
```

Use the following command to verify the new server certificate is valid and trusted:

```
$ certutil -V -u V -d /usr/local/apache2/alias -n Server-Cert
```

- `certutil: certificate is valid`

Server NSS Configuration Settings

The following section involves configuring the Apache web server settings by editing the NSS configuration file (nss.conf). By default the file is located in `/usr/local/apache2/conf/extra/`.

1) First verify that the server will load the nss.conf file. Open the httpd.conf file with:

```
$ vi /usr/local/apache2/conf/httpd.conf
```

Include the following line to specify the path of the nss.conf file under the Supplemental configuration section (if not already specified):

```
Include conf/nss.conf
```

If the target server is one that has mod_ssl on it, comment the line that includes the mod_ssl configuration file.

```
#Include conf/extra/httpd-ssl.conf
```

2) Open the nss.conf file. You may use the vi text editor:

   ```
   $ vi /usr/local/apache2/conf/nss.conf
   ```

   Type *i* to trigger insert mode.

3) Find the following setting and ensure it is uncommented and matches the following:

   ```
   Listen 443
   <VirtualHost _default_:443>
   ```

4) Find the NSSVerifyClient property and enable client authentication by setting it to **require**.

   ```
   NSSVerifyClient require
   ```

5) Ensure the server certificate is specified. In previous steps, the server certificate was given the nickname `Server-Cert`. Ensure the NSSNickname directive specifies `Server-Cert:`

   ```
   NSSNickname Server-Cert
   ```

6) Find the NSSProtocol property. According to NIST, TLS protocol 1.2 or higher should be used. TLS Version 1.0 or 1.1 should only be used if required for non-government agency interoperability. NSS handles the specified protocols as "ranges", and automatically negotiates the use of the strongest protocol for a connection starting with the maximum specified protocol and downgrading as necessary to the minimum specified protocol that can be used between two processes. Specify the following:

   ```
   NSSProtocol TLSv1.0,TLSv1.1,TLSv1.2
   ```

7) Find the DocumentRoot property and specify the root directory of your webpage documents (html home page, files, etc):

   ```
   DocumentRoot "<your_docroot_location>"
   ```

8) Add the NSSFIPS property in the **SSL Virtual Host Context** and set it to "on" to turn on FIPS mode:

   ```
   NSSFIPS on
   ```

9) Find the NSSOCSP property and turn on OCSP validation. OCSP will use the AIA OCSP value in the certificate to locate the OCSP responder to be used:

   ```
   NSSOCSP on
   ```

10) Be sure the ErrorLog and TransferLog lines are uncommented and confirm the log locations.

11) Comment out the existing NSSCipherSuite property by putting a pound sign (#) at the beginning of the line, and add the following line:

```
NSSCipherSuite
+ecdhe_rsa_3des_sha,+ecdhe_rsa_aes_128_sha,+ecdhe_rsa_aes_128_sha_256,+ecdhe_rs
a_aes_128_gcm_sha_256,-rsa_rc2_40_md5,-rsa_rc4_40_md5,-rsa_des_56_sha,-
rsa_rc4_56_sha,-rsa_rc4_128_md5,-rsa_rc4_128_sha,-rsa_null_md5
```

Note: This text should all be on a single line, although it is shown wrapped to several.

12) Save the file by pressing the ESC button, typing *:wq* and pressing Enter.

13) Configure the HTTP server to not listen on port 80. Comment out the following flag in the */usr/local/apache2/conf/httpd*.conf file (use the 'vi' text editor in the same manner as in the previous step):

```
# Listen 80
```

Revocation Checking Configuration

NSS certificate validation can be configured to do OCSP or CRL checking. When both are implemented, CRL checking is primary and OCSP is secondary. Apache will attempt CRL checking first and then failover to OCSP checking. There is an OCSP flag that can be set to "On", which is mentioned in the **Server NSS Configuration Settings** section above.

This section will provide the steps required to properly acquire and install the CA CRLs. In addition, it will provide steps on setting up a cronjob under the HTTPD service user account that will download and install the CRLs on a nightly basis.

NOTE: It is extremely important that the CRLs are installed and maintained in a secure fashion. Apache HTTP server by default will allow access to users with revoked certificates if the corresponding CRL is not installed. In this event, the server will skip revocation checking. CRLs MUST be installed and refreshed daily using an automated process. The script that is provided below for initial installation may be modified and scheduled to run via cron.

1) Make a directory for storage of the CRLs:

```
$ mkdir /usr/local/apache2/crls
```

2) Determine the HTTPD service user/group. Open the httpd.conf file:

```
$ vi /usr/local/apache2/conf/httpd.conf
```

Verify the name listed for the user and group.

```
User apache

Group apache
```

This will be the name that should be specified in place of <apache_user> in the following steps. Close the file by first pressing ESC and then entering *:q* and pressing Enter.

3) Give the HTTPD service user ownership of the /usr/local/apache2/crls directory created in step 1. Replace the bracketed text with the correct HTTPD service user:

```
$ chown <apache_user> /usr/local/apache2/crls
```

4) Download the latest version of the CRLAutoCache for Linux:

Unclassified/NIPRNet Systems:

The **NIPRNet CRLAutoCache for Linux** tool can be downloaded from the DoD PKE website at http://iase.disa.mil/pki-pke under **Tools > Certificate Validation**. Select the **NIPRNet CRLAutoCache for Linux** link. Save the package and uncompress the folder into the /usr/local/apache2/crls folder. Rename this folder to CRLAutoCache_Linux.

Secret/SIPRNet Systems:

The script can be downloaded from the DoD PKE website at http://iase.disa.smil.mil/pki-pke under **Tools > Certificate Validation**. Select the **SIPRNet CRLAutoCache for Linux** link. Save the package and uncompress the folder into the /usr/local/apache2/crls folder. Rename this folder to CRLAutoCache_Linux.

NOTE: Refer to the CRLAutoCache for Linux User Guide for an overview of the script. The document contains guidance on how to use the script and the options associated with it. This document can be found on http://iase.disa.mil/pki-pke (NIPRNet) or http://iase.disa.smil.mil/pki-pke (SIPRNet). It will be in the same download location as the tool.

5) Change the ownership of CRLAutoCache_Linux.sh in the folder to the HTTPD service user and run the script as the HTTPD service user. Replace the bracketed text with the correct HTTPD service user:

```
$ chown <apache_user> /usr/local/apache2/crls/CRLAutoCache_Linux/
$ cd /usr/local/apache2/crls/CRLAutoCache_Linux/
$ chown <apache_user> CRLAutoCache_Linux.sh
```

6) Run the CRLAutoCache_Linux.sh script as the apache user using the following command. Replace the bracketed value with the correct HTTPD service user (the following is a NIPR example):

NOTE: Before running this command, give the Apache user persmission to write to the NSS database directory with: `chmod 775 /usr/local/apache/alias.`

NOTE: Before running this command, be sure to refer to the CRLAutoCache for Linux guide for guidance on configuring the NSS database password. The password is required in order to install the CRLs into the database if using a later version of NSS. The apache_user must have read permssions on the nssdb password file. Refer to the guide for more details.

```
$ su <apache_user>  -s /bin/bash -c \
"/usr/local/apache2/crls/CRLAutoCache_Linux/CRLAutoCache_Linux.sh --dest \
/usr/local/apache2/crls/ --ECA /usr/local/apache2/crls/ --nss
/usr/local/apache2/alias"
```

7) Confirm the CRLs have been installed using the following command:

```
$ crlutil -L -d /usr/local/apache2/alias
```

8) Schedule the CRLAutoCache_Linux.sh script to run nightly. The following cronjob will run the script nightly and place the output in a log file. Since this cronjob will be run daily with minimal monitoring, it should be run under the HTTPD user account that has ownership of the /usr/local/apache2/crls/ folder. Use the following command to open the cronjobs for the HTTPD service user account in a text editor. Replace the bracketed value with the correct HTTPD service user.

```
$ crontab -e -u <apache_user>
```

Type *i* to insert text into the file. Add the following two lines into the cron file to schedule the CRLAutoCache_Linux.sh script to run every day at 3 a.m. (see note below):

```
00 3 * * * /usr/local/apache2/crls/CRLAutoCache_Linux/CRLAutoCache_Linux.sh \

--dest/usr/local/apache2/crls/ --nss /usr/local/apache2/alias >> \
/var/log/CRLAutoCache_Linux.log 2>&1
```

Save and exit the crontab file by pressing the ESC button, typing *:wq* and then pressing Enter.

NOTE: It is extremely important to verify the CRLs are up-to-date; if the CRLs are expired, mod_nss will deny access to all clients with certificates from the CA with the expired CRL. It is recommended to download every day during non-business hours. 3 a.m. is a suggested time, but the time can be adjusted as long as it is not during peak business/network traffic hours.

9) After the CRLAutoCache_Linux.sh script is run, the HTTPD service will need to be reloaded. Services can only be reloaded as a root user. Create the following cronjob for the root user. This cronjob will reload the HTTPD service every day an hour after the CRLAutoCache_Linux.sh script is run. This ensures the CRLs are properly loaded after being downloaded. Output will be placed in a log for verification.

Use the following command to open the cronjobs for the root user account in a text editor:

```
$ crontab -e
```

Type *i* to insert text into the file. Add the following text into the cron file to schedule the HTTPD service to run every day at 4:00 a.m. (time may vary according to your setup, but should be at least 1 hour following the scheduled CRL download time):

```
00 4 * * * /usr/local/apache2/bin/apachectl restart >>
/var/log/httpd_restart.log 2>&1
```

Save and exit the 'crontab' file by pressing the ESC button, typing *:wq* and pressing Enter.

Auditing/Logging

Keeping track of the users who are authenticating to the web server is an important element of keeping the server secure. The following instructions describe how to configure the server to capture the certificate Distinguished Name and other information for each client who attempts to authenticate to the server.

1) Create a custom log file in the /usr/local/apache2/logs/ directory.

   ```
   $ vi /usr/local/apache2/logs/client_access_log
   ```

   Save the file by pressing the ESC button, typing *:wq* and pressing Enter.

2) Open the nss configuration file.

   ```
   $ vi /usr/local/apache2/conf/nss.conf
   ```

3) Add the following line under the <VirtualHost_default_:443> section right below the rest of the logging directives :

   ```
   CustomLog logs/client_access_log "%h %l %u %t %{SSL_CLIENT_I_DN}x
   \%{SSL_CLIENT_S_DN}x"
   ```

   Save the file by pressing the ESC button, typing *:wq* and pressing Enter.

Now when a user attempts to authenticate to the web server, the server will log the user's IP address, the date and time of authentication, and the issuer and subject of the client certificate.

NOTE: Other logging directives should be configured in accordance with the official Apache STIG and your organizational requirements.

Start the Server

Start the server with the following command. You will be prompted to enter the NSS database password that was set previously.

```
$ /usr/local/apache2/bin/apachectl start
```

NOTE: If you receive an error about the NSSPassPhraseHelper similar to:

```
"NSSPassPhraseHelper: /usr/libexec/nss_pcache does not exist or is not executable"
```

Apache may be looking for the nss_pcache executable in the wrong long location.

Locate the correct location of the nss_pcache file (may be in the mod_nss source folder) and specify it on `NSSPassPhraseHelper` **line in the nss.conf file**.

Test the server by navigating to https://<server_FQDN> in a web browser.

NOTE: Make sure to open any host or external firewalls for inbound port 443.

NOTE: You will need a DoD-issued client certificate available (e.g. from your CAC) to test client certificate authentication to the server.

Configure Apache HTTP with mod_ssl

The following procedures detail the steps to public key enable an Apache HTTP web server with mod_ssl [7] in FIPS mode.

FIPS Mode

In order to enable mod_ssl in FIPS mode, the OpenSSL FIPS Object Module must be compiled to integrate with OpenSSL and the httpd service. The following section will guide a user through the process of compiling the OpenSSL FIPS Object Module and linking it with OpenSSL.

1) For FIPS installation instructions, refer to the FIPS 140-2 Object Module User Guide v2.0. Thoroughly read this guide and follow the steps specified to compile and install the module. The document is available at http://www.openssl.org/docs/fips/UserGuide-2.0.pdf.
2) Refer to **Appendix C: Building OpenSSL and httpd in FIPS mode** for further instructions after building the FIPS 140-2 Object Module.

Installing mod_ssl

The mod_ssl module should be enabled during the httpd build process. The `--enable-ssl` flag should be specified when running the configure script.

Trusted Certificate Configuration

1) Create the folder where the CA certificates will be stored with the command.

```
$ mkdir -p /usr/local/apache2/certs/ca_certs/
```

2) Navigate to the CA certificates folder that was created.

```
$ cd /usr/local/apache2/certs/ca_certs/
```

3) Obtain the PKI CA certificates for installation into the server database.

Unclassified/NIPRNet Systems:

The NIPRNet DoD PKI CA certificates can be obtained as PKCS#7 files from the DoD PKE Engineering web site at http://iase.disa.mil/pki-pke under **Tools > Trust Store**. Select the appropriate certificate bundle under **PKI CA Certificate Bundles: PKCS#7.** Save the package locally. Extract the zip file and navigate into the extracted directory structure.

Follow the instructions in the README.txt to validate the signature of the PKCS#7 package.

Secret/SIPRNet Systems:

The SIPRNet/NSS PKI CA certificates can be obtained as PKCS#7 files from the SIPR DoD PKE Engineering web site at http://iase.disa.smil.mil/pki-pke under **Tools > Trust Store**. Select the appropriate certificate bundle under **PKI CA Certificate Bundles: PKCS#7**. Save the package locally. Extract the zip file and navigate into the extracted directory structure.

Follow the instructions in the README.txt to validate the signature of the PKCS#7 package.

4) Create a system wide environment variable OPENSSL_FIPS=1. There are numerous ways this can be done such as editing the file /etc/profile adding the line "export OPENSSL_FIPS=1" to the file.

   After adding the environment variable logout of the system and then log back in to confirm the environment variable is set. Execute "env |grep OPENSSL_FIPS" and verify the line "OPENSSL_FIPS=1" appears in the output.

5) Convert the pem-signed .p7b file(s) to a single PEM file of concatenated certificates using 'openssl' by entering the following command(s) from the directory containing the p7b file(s) (file names will vary and "\" indicates there is a space between arguments):

   Unclassified/NIPRNet Systems:

```
$ openssl pkcs7 -inform PEM -outform PEM -in\

Certificates_PKCS7_v4.0.1_DoD.pem-signed.p7b -out\
/usr/local/apache2/certs/ca_certs/alldodcerts.pem -print_certs
```

   Secret/SIPRNet Systems:

```
$ openssl pkcs7 -inform PEM -outform PEM -in\

Certificates_PKCS7_v4.0.1_NSS.pem-signed.p7b -out\
/usr/local/apache2/certs/ca_certs/allNSScerts.pem -print_certs
```

   NOTE: It is extremely important to verify the authenticity of the PKCS#7 file signature prior to installation.

Create the Private Key and Certificate Signing Request

1) If it does not already exist, create the folder `/usr/local/apache2/private` for storage of the private key with the following command:

```
$ mkdir -p /usr/local/apache2/private/
```

2) Create a 2048-bit RSA private key and Certificate Signing Request (CSR) for the Apache server. This key will be Triple DES encrypted and PEM formatted. Use the command:

NOTE: Use the 'dod-openssl.cfg' file in Appendix B: dod-openssl.cfg. Save this file in the /usr/local/apache2/ directory and specify it according to the command below.

```
$ openssl req -new -out /usr/local/apache2/private/dodserver-pkcs10.csr -newkey
rsa:2048 -keyout /usr/local/apache2/private/dodserverkey.key -config
/usr/local/apache2/dod-openssl.cfg
```

Unclassified/NIPRNet Systems:

Enter the following values when prompted:

a. **Enter PEM pass phrase:** *Enter a passphrase*

   NOTE: The PEM pass phrase will be required when starting the server

b. **Verifying – Enter PEM pass phrase:** *Enter passphrase set in previous step*

c. **Country Name** – *Press Enter to accept default (US)*

d. **Organization Name** - *Press Enter to accept default (U.S. Government)*

e. **Organizational Unit Name** - *Press Enter to accept default (DoD)*

f. **Organizational Unit Name** - *Press Enter to accept default (PKI)*

g. **Organizational Unit Name** - *Enter your specific Organizational Unit (e.g. USA, USAF, DISA)*

h. **Common Name** – *Enter your web server Fully Qualified Domain Name*

Secret/SIPRNet Systems:

Enter the following values when prompted:

a. **Enter PEM pass phrase:** *Enter passphrase*

   NOTE: The PEM pass phrase will be required when starting the server

b. **Verifying – Enter PEM pass phrase:** *Enter passphrase set in previous step*

c. **Country Name** – *Press Enter to accept default (US)*

d. **Organization Name** - *Press Enter to accept default (U.S. Government)*

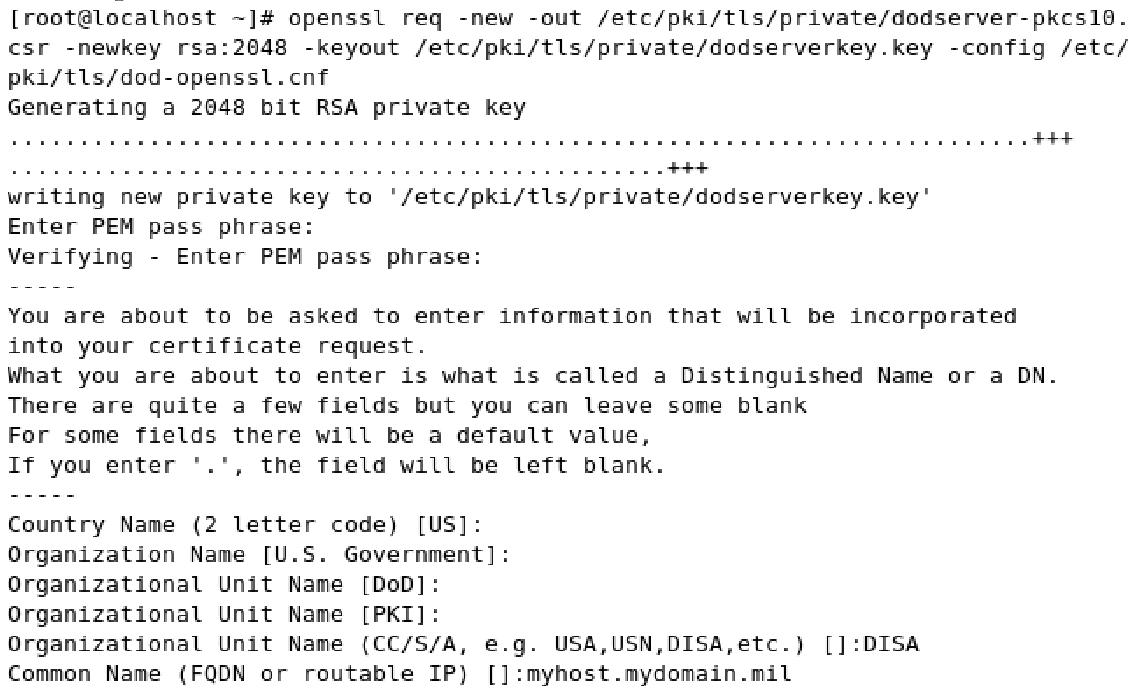
i. **Organizational Unit Name** - *Press Enter to accept default (NSS)*

e. **Organizational Unit Name** - *Press Enter to accept default (DoD)*

f. **Organizational Unit Name** - *Enter your specific Organizational Unit (e.g. USA, USAF, DISA)*

g. **Common Name** – *Enter your web server Fully Qualified Domain Name*

The output should look similar to the screenshot below:

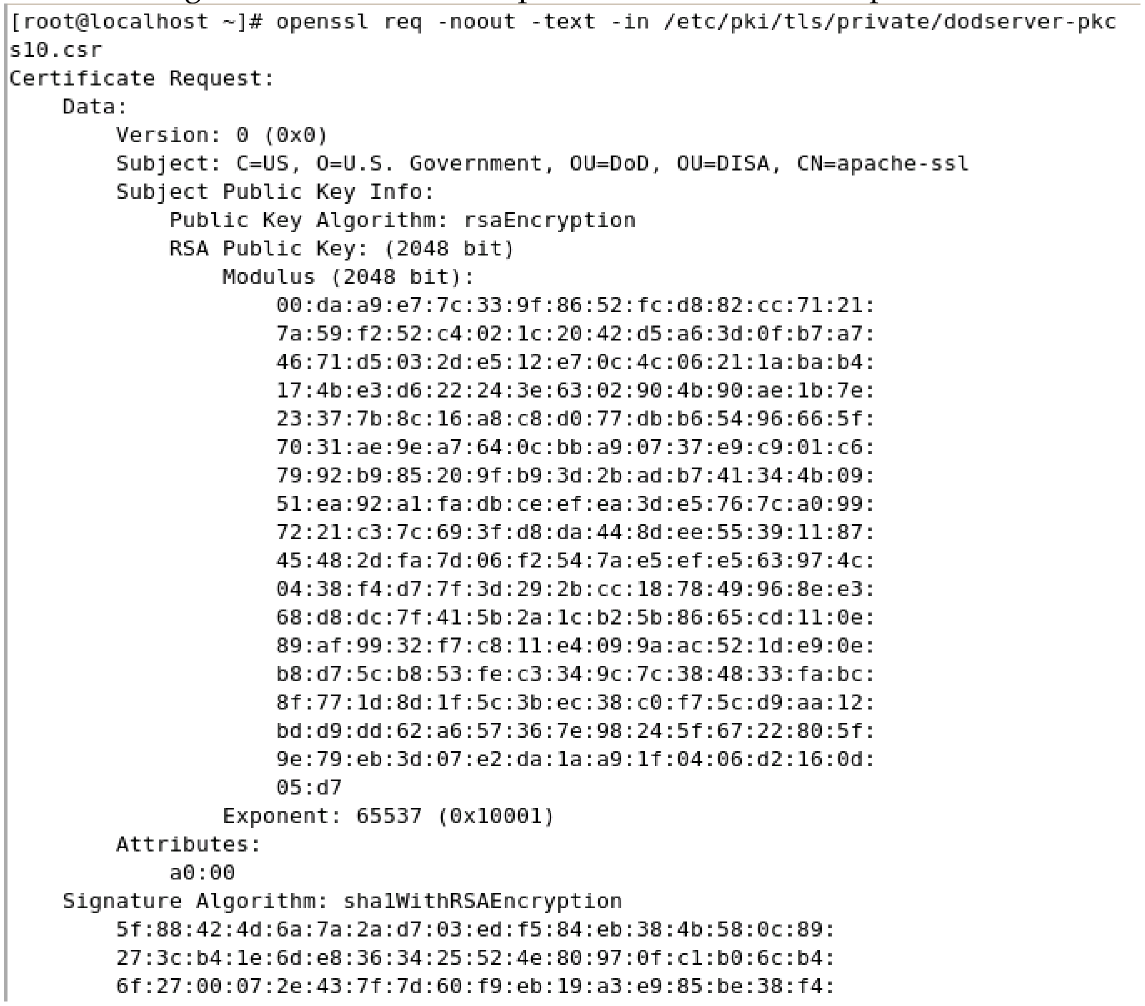


```
[root@localhost ~]# openssl req -new -out /etc/pki/tls/private/dodserver-pkcs10.
csr -newkey rsa:2048 -keyout /etc/pki/tls/private/dodserverkey.key -config /etc/
pki/tls/dod-openssl.cnf
Generating a 2048 bit RSA private key
.......................................................................+++
...........................................+++
writing new private key to '/etc/pki/tls/private/dodserverkey.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
Organization Name [U.S. Government]:
Organizational Unit Name [DoD]:
Organizational Unit Name [PKI]:
Organizational Unit Name (CC/S/A, e.g. USA,USN,DISA,etc.) []:DISA
Common Name (FQDN or routable IP) []:myhost.mydomain.mil
```

The details of the CSR can be displayed with the following command:

```
$ openssl req -noout -text -in /usr/local/apache2/private/dodserver-pkcs10.csr
```

The following screenshot is an example of a CSR detailed output:



```
[root@localhost ~]# openssl req -noout -text -in /etc/pki/tls/private/dodserver-pkc
s10.csr
Certificate Request:
    Data:
        Version: 0 (0x0)
        Subject: C=US, O=U.S. Government, OU=DoD, OU=DISA, CN=apache-ssl
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:da:a9:e7:7c:33:9f:86:52:fc:d8:82:cc:71:21:
                    7a:59:f2:52:c4:02:1c:20:42:d5:a6:3d:0f:b7:a7:
                    46:71:d5:03:2d:e5:12:e7:0c:4c:06:21:1a:ba:b4:
                    17:4b:e3:d6:22:24:3e:63:02:90:4b:90:ae:1b:7e:
                    23:37:7b:8c:16:a8:c8:d0:77:db:b6:54:96:66:5f:
                    70:31:ae:9e:a7:64:0c:bb:a9:07:37:e9:c9:01:c6:
                    79:92:b9:85:20:9f:b9:3d:2b:ad:b7:41:34:4b:09:
                    51:ea:92:a1:fa:db:ce:ef:ea:3d:e5:76:7c:a0:99:
                    72:21:c3:7c:69:3f:d8:da:44:8d:ee:55:39:11:87:
                    45:48:2d:fa:7d:06:f2:54:7a:e5:ef:e5:63:97:4c:
                    04:38:f4:d7:7f:3d:29:2b:cc:18:78:49:96:8e:e3:
                    68:d8:dc:7f:41:5b:2a:1c:b2:5b:86:65:cd:11:0e:
                    89:af:99:32:f7:c8:11:e4:09:9a:ac:52:1d:e9:0e:
                    b8:d7:5c:b8:53:fe:c3:34:9c:7c:38:48:33:fa:bc:
                    8f:77:1d:8d:1f:5c:3b:ec:38:c0:f7:5c:d9:aa:12:
                    bd:d9:dd:62:a6:57:36:7e:98:24:5f:67:22:80:5f:
                    9e:79:eb:3d:07:e2:da:1a:a9:1f:04:06:d2:16:0d:
                    05:d7
                Exponent: 65537 (0x10001)
        Attributes:
            a0:00
    Signature Algorithm: sha1WithRSAEncryption
        5f:88:42:4d:6a:7a:2a:d7:03:ed:f5:84:eb:38:4b:58:0c:89:
        27:3c:b4:1e:6d:e8:36:34:25:52:4e:80:97:0f:c1:b0:6c:b4:
        6f:27:00:07:2e:43:7f:7d:60:f9:eb:19:a3:e9:85:be:38:f4:
```

3) Display the .csr file in PEM format using the "cat" command below:

```
$ cat /usr/local/apache2/private/dodserver-pkcs10.csr
```

Copy the request including the "-----BEGIN CERTIFICATE REQUEST-----" and "-----END CERTIFICATE REQUEST-----". This will be pasted into the certificate request submission form at the CA website.

5) The next step is to submit the request. For NIPRNet, instructions on how to submit a SSL server certificate request can be obtained from the DoD PKE web site at http://iase.disa.mil/pki-pke. For SIPRNet, instructions on how to submit a SSL server certificate request can be obtained from the SIPRNet DoD PKE Web site at http://iase.disa.smil.mil/pki-pke/ . Navigate to **For Administrators, Integrators & Developers > Web Servers** and select the guide **Obtaining a PKI Certificate for a DoD Server**. Use this guide to submit and retrieve the request generated above.

Additional Action: The above guide will direct users to retrieve the **Base 64 encoded certificate**. In addition to this, retrieve the **Base 64 encoded certificate with CA certificate chain in pkcs7 format**. Name this file "`dodservercertchain.p7b`" and save it to "`/usr/local/apache2/certs/dodservercertchain.p7b`". Convert this file to '.pem' format using the following command:

```
$ openssl pkcs7 -inform PEM -outform PEM -in dodservercertchain.p7b -out \
dodservercertchain.pem -print_certs
```

This file will be used in **Server SSL Configuration Settings**.

6) Once you have retrieved your certificate(Base 64 encoded certificate), name it `dodservercert.cer`. Save the certificate to `/usr/local/apache2/certs/`.

Server SSL Configuration Settings

The following section involves configuring the Apache web server settings by editing the mod_ssl configuration file (httpd-ssl.conf). Instructions from this point on will assume that a FIPS-compatible version of httpd has been built according to **Appendix C: Building OpenSSL and httpd in FIPS mode** and installed to /usr/local/apache2. If the install directory of httpd is different, modify the command paths accordingly.

1) First verify that the server will load the http-ssl.conf file. Open the httpd.conf file with:

```
$ vi /usr/local/apache2/conf/httpd.conf
```

Remove the # in front of the line.

```
#Include conf/extra/httpd-ssl.conf
```

Be sure the following lines are uncommented as well to ensure the server is properly configure for SSL:

```
LoadModule ssl_module modules/mod_ssl.so
LoadModule socache_shmcb_module modules/mod_socache_shmcb.so
```

Comment out the directive for the server to listen on port 80.

```
# Listen 80
```

Save and exit the file by pressing the ESC button, typing *:wq* and pressing Enter.

2) Open the httpd-ssl.conf file:

```
$ vi /usr/local/apache2/conf/extra/httpd-ssl.conf
```

3) Type *i* to edit the file. Remove or comment out the default 'SSLRandomSeed' and add the 'SSLRandomSeed' values below for the startup file and connect file.

```
#SSLRandomSeed startup file:/dev/random 512
#SSLRandomSeed startup file:/dev/urandom 512
#SSLRandomSeed connect file:/dev/random 512
```

```
#SSLRandomSeed connect file:/dev/urandom 512
SSLRandomSeed startup file:/dev/urandom 2048
SSLRandomSeed connect file:/dev/urandom 2048
```

4) Verify SSLEngine is set to **on**. If SSLEngine is set to off, then remove or comment out the 'SSLEngine off' line and add 'SSLEngine on'.

```
# SSL Engine Switch:
# Enable/Disable SSL for this virtual host.
#SSLEngine off
SSLEngine on
```

5) Remove or comment out the default SSLCipherSuite line and add the new SSLCipherSuite line below to disable unacceptable cipher suites (there is a space after SSLCipherSuite, the cipher list is one line):

```
# SSL Cipher Suite:
# List the ciphers that the client is permitted to negotiate.
# See the mod_ssl documentation for a complete list.
#SSLCipherSuite ALL:!ADH:!EXPORT:!SSLv2:RC4+RSA:+HIGH:+MEDIUM:+LOW
SSLCipherSuite DES-CBC3-SHA:AES128-SHA:AES256-SHA:ECDHE-ECDSA-DES-CBC3-
SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-DES-CBC3-SHA:ECDHE-RSA-AES128-SHA:AES256-
GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES256-GCM-SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES128-GCM-
SHA256:!aNULL:!eNULL:!LOW:!3DES:!MD5:!EXP:!PSK:!DSS:!RC4:!SEED:!EXPORT:!EXP
```

6) Remove or comment out the default server certificate and add the DoD-issued server certificate that was retrieved. Do this by specifying the file path of the server certificate.

```
# Server Certificate:
# Point SSLCertificateFile at a PEM encoded certificate. If
# the certificate is encrypted, then you will be prompted for a
# pass phrase. Note that a kill -HUP will prompt again. A new
# certificate can be generated using the genkey(1) command.
#SSLCertificateFile "/usr/local/apache2/certs/localhost.crt"
SSLCertificateFile "/usr/local/apache2/certs/dodservercert.cer"
```

7) Remove or comment out the default server certificate private key and add the DoD server certificate private key path.

```
# Server Private Key:
# If the key is not combined with the certificate, use this
# directive to point at the key file. Keep in mind that if
# you've both a RSA and a DSA private key you can configure
# both in parallel (to also allow the use of DSA ciphers, etc.)
#SSLCertificateKeyFile "/usr/local/apache2/private/localhost.key"
```

```
SSLCertificateKeyFile "/usr/local/apache2/private/dodserverkey.key"
```

8) Remove the default server certificate chain path and add the following:

```
# Server Certificate Chain:
# Point SSLCertificateChainFile at a file containing the
# concatenation of PEM encoded CA certificates which form the
# certificate chain for the server certificate. Alternatively
# the referenced file can be the same as SSLCertificateFile
# when the CA certificates are directly appended to the server
# certificate for convinience.
#SSLCertificateChainFile "/usr/local/apache2/certs/server-chain.crt"
SSLCertificateChainFile "/usr/local/apache2/certs/dodservercertchain.pem"
```

9) Remove or comment out the default Certification Authority (CA) certificate file, if it was not already, and add the CA certificates PEM file that was generated in the **Create the Private Key and Certificate Signing Request** section.

```
# Certificate Authority (CA):
# Set the CA certificate verification path where to find CA
# certificates for client authentication or alternatively one
# huge file containing all of them (file must be PEM encoded)
#SSLCACertificateFile "/usr/local/apache2/certs/ca-bundle.crt"
SSLCACertificateFile "/usr/local/apache2/certs/ca_certs/alldodcerts.pem"
```

10) Under the Certificate Revocation Lists (CRL) section, add the revocation path /usr/local/apache2/crls/. Also add the SSLCARevocationCheck directive with the `chain` flag to indicate the server should check revocation of all the issuing certificates in the end-entity certificate chain.

```
#   Certificate Revocation Lists (CRL):
#   Set the CA revocation path where to find CA CRLs for client
#   authentication or alternatively one huge file containing all
#   of them (file must be PEM encoded).
#   The CRL checking mode needs to be configured explicitly
#   through SSLCARevocationCheck (defaults to "none" otherwise).
#   Note: Inside SSLCARevocationPath you need hash symlinks
#         to point to the certificate files. Use the provided
#         Makefile to update the hash symlinks after changes.
SSLCARevocationPath "/usr/local/apache2/crls/"
SSLCARevocationCheck chain
```

NOTE: The CRL directory will be created in the next section.

11) Change the SSLVerifyClient value to **require** users to present certificates to connect, and verify that the SSLVerifyDepth value is 3 or higher. This setting determines how many issuing certificates can be in a validation path. Typical DoD certificates only have 2 issuing certificates(the intermediate and the root) in a direct trust path. Be sure to accomadate all PKI trust chains (DoD and/or non-DoD) that will be validated by the server.

```
# Client Authentication (Type):
# Client certificate verification type and depth. Types are
# none, optional, require and optional_no_ca. Depth is a
# number which specifies how deeply to verify the certificate
# issuer chain before deciding the certificate is not valid.
SSLVerifyClient require
SSLVerifyDepth 10
```

12) Specify the SSLProtocols in which the server shold use by placing the following line under the Client Authentication (Type) section from the previous step. According to NIST, TLS protocol 1.2 or higher should be used. TLS Version 1.0 or 1.1 should only be used if required for non-government agency interoperability. TLSv1.1 and TLSv1.2 can be used only if the OpenSSL version is 1.0.1 or above.

```
SSLProtocol +TLSv1 +TLSv1.1 +TLSv1.2
```

13) Verify one of the SSLOptions for “SSL Global Context” is **+StrictRequire**. If +StrictRequire is not present then add it.

```
#SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
#<Files ~ "\.(cgi|shtml|phtml|php3?)$">
# SSLOptions +StdEnvVars
#</Files>
#<Directory "/var/www/cgi-bin">
# SSLOptions +StdEnvVars
#</Directory>
SSLOptions +StrictRequire
```

14) Place the following line in the file under the “SSL Global Context” section to turn on FIPS mode:

```
SSLFIPS on
```

15) Save the changes by pressing the ESC button, typing *:wq* and pressing Enter.

Revocation Checking Configuration

Certificate validation for a mod_ssl server can be performed with either OCSP (available with HTTPD 2.3 and later, if using OpenSSL 0.9.7 or later) or CRL checking. The following section provides instructions on how to configure OCSP and CRL validation. In addition, it provides guidance for setting up a cronjob under the HTTPD service user account that will download and install the CRLs on a nightly basis.

NOTE: It is critical that certificate validation is configured correctly. We recommend configuring either OCSP alone or CRL checking alone. This is because in the functionality of Apache 2.4, if both CRL checking and OCSP checking are configured at the same time, the server will attempt to validate via both sources. This creates unneccesary traffic when a certificate has already been successfully validated by one of the protocols. CRL checking is recommended for servers that will be authenticating a high volume of users. If CRLs are being used, it is extremely important that the CRLs are installed and maintained in a secure fashion. CRLs MUST be installed and refreshed daily using an automated process. The script that is provided below for initial installation may be modified and scheduled to run via cron jobs.

OCSP Configuration

NOTE: If OCSP is being configured, turn off CRL checking by commenting out the following lines from the httpd-ssl.conf document:

```
#SSLCARevocationPath "/usr/local/apache2/crls/"
#SSLCARevocationCheck chain
```

1) Add the SSLOCSPEnable flag to the httpd-ssl.conf file. Place the following line in the file under the <VirtualHost….> section:

   ```
   SSLOCSPEnable on
   ```

   If the SSLOCSPEnable flag alone is enabled, the server will look in the Authority Information Access extension of the certificate to determine the OCSP responder URL to use.

CRL Configuration

NOTE: If CRLs are being configured, turn off OCSP checking by commenting out the following line from the httpd-ssl.conf document:

```
#SSLOCSPEnable on
```

1) Create a folder which will store the CRLs. Use the following:

   ```
   $ mkdir -p /usr/local/apache2/crls/
   ```

2) Determine the HTTPD service user/group. Open the `httpd.conf` file:

   ```
   $ vi /usr/local/apache2/conf/httpd.conf
   ```

   Verify the name listed for the user and group.

   ```
    User apache
    Group apache
   ```

This will be the name that should be specified in place of <apache_user> in the following steps. Close the file by pressing the ESC button, typing *:q* and pressing Enter.

3) Change the ownership of both folders to the Apache HTTPD service user. The scheduled cronjob for the CRLAutoCache_Linux.sh script below will need to be run as the HTTPD service user for security purposes. More details are in the following steps. Replace the bracketed text with the HTTPD service user identified in step 3.

```
$ chown <apache_user> /usr/local/apache2/crls/
```

4) Download the latest version of the CRLAutoCache for Linux:

Unclassified/NIPRNet Systems:

The **NIPRNet CRLAutoCache for Linux** tool can be downloaded from the DoD PKE website at http://iase.disa.mil/pki-pke under **Tools > Certificate Validation**. Select the **NIPRNet CRLAutoCache for Linux** link. Save the package to the folder `/usr/local/apache2/crls`. Rename this folder to CRLAutoCache_Linux.

Secret/SIPRNet Systems:

The script can be downloaded from the DoD PKE website at http://iase.disa.smil.mil/pki-pke under **Tools > Certificate Validation**. Select the **SIPRNet CRLAutoCache for Linux** link. Save the package to the folder `/usr/local/apache2/crls`. Rename this folder to CRLAutoCache_Linux.

NOTE: Refer to the CRLAutoCache Linux User Guide for an overview of the script. The document contains guidance on how to use the script and the options associated with it. This document can be found on http://iase.disa.mil/pki-pke (NIPRNet) or http://iase.disa.smil.mil/pki-pke (SIPRNet). It will be in the same download location as the tool.

5) Change the owner of the script so that the HTTPD service user may run the script. Replace the bracketed text with the HTTPD service user:

```
$ chown <apache_user> /usr/local/apache2/crls/CRLAutoCache_Linux/

$ cd /usr/local/apache2/crls/CRLAutoCache_Linux

$ chown <apache_user> CRLAutoCache_Linux.sh
```

6) Run the CRLAutoCache_Linux.sh script as the HTTPD service user using the following command. Replace the bracketed text with the HTTPD service user (the following is a NIPR example):

```
$ su <apache_user> -s /bin/bash -c \
"/usr/local/apache2/crls/CRLAutoCache_Linux/CRLAutoCache_Linux.sh --dest \
/usr/local/apache2/crls/ --ECA /usr/local/apache2/crls/ --openssl
/usr/local/apache2/crls/"
```

The script will do the following things:

- Download the CRLs from the Global Directory Service
- Unzip and place all the CRLs in the specified folder (for use with the 'SSLCARevocationPath' directive)
- Generate hashes of each CRL (for use with the 'SSLCARevocationPath' directive)

List the contents of the /usr/local/apache2/crls/ directory to confirm the script generated all the related files.

7) Schedule the CRLAutoCache_Linux.sh script to run nightly. The following cronjob will run the script nightly and place the output in a log file. Since this cronjob will be run daily with minimal monitoring, it should be run under the Apache HTTPD user account that has ownership of the /usr/local/apache2/crls/ folder. Use the following command to open the cronjobs for the HTTPD user account in a text editor. Replace the bracketed text with the correct HTTPD service user:

```
$ crontab -e -u <apache_user>
```

Type *i* to insert text into the file. Add the following two lines into the cron file to schedule the CRLAutoCache_Linux.sh script to run every day at 3 a.m. (see note below):

```
00 3 * * * /usr/local/apache2/crls/CRLAutoCache_Linux/CRLAutoCache_Linux.sh --
dest \ /usr/local/apache2/crls/ --ECA /usr/local/apache2/crls --openssl
/usr/local/apache2/crls/ >> \ /var/log/CRLAutoCache_Linux.log 2>&1
```

Save and exit the **crontab** file by pressing the ESC button, typing *:wq* and then pressing Enter.

NOTE: It is extremely important to verify the CRLs are up-to-date. It is recommended to download every day during non-business hours. 3 a.m. is not a required time; the time can be adjusted as long as it is not during peak business/network traffic hours.

8) After the CRLAutoCache_Linux.sh script is run the HTTPD service will need to be reloaded. Services can only be reloaded as a root user. Create the following cronjob for the root user. This cronjob will reload the HTTPD service every day an hour after the CRLAutoCache_Linux.sh script is run. This ensures the CRLs are properly loaded after being downloaded. Output will be placed in a log for verifications.

Use the following command to open the cronjobs for the root user account in a text editor:

```
$ crontab -e
```

Type *i* to insert text into the file. Add the following text into the cron file to schedule the HTTPD service to run every day at 4:00 a.m. (time may vary according to your setup, but should be at least an hour following the scheduled CRL download time):

```
00 4 * * * /sbin/service httpd reload >> /var/log/httpd_reload.log 2>&1
```

Save and exit the 'crontab' file by pressing the ESC button, typing *:wq* and pressing Enter.

Auditing/Logging

Keeping track of the users who are logging into the web server is an important element of keeping the server secure. The following instructions walk through the steps required to configure the server to capture the certificate Distinguished Name and other information for each client who attempts to authenticate to the server.

1) Create a custom log file in the /usr/local/apache2/logs directory.

```
$ touch /usr/local/apache2/logs/client_access_log
```

Save the file by pressing the ESC button, typing *:wq* and pressing Enter.

2) Open the mod_ssl configuration file.

```
$ vi /usr/local/apache2/conf/extra/httpd-ssl.conf
```

3) Add the following line under the <VirtualHost_default_:443> section right below the rest of the logging directives:

```
CustomLog logs/client_access_log "%h %l %u %t %{SSL_CLIENT_I_DN}x \
%{SSL_CLIENT_S_DN}x"
```

Save the file by pressing the ESC button, typing *:wq* and pressing Enter.

4) Now when a user attempts to authenticate to the web server, the server will log the user's IP address, the date and time of authentication, and the issuer and subject of the client certificate.

NOTE: Other logging directives should be configured in accordance with the official Apache STIG and your organizational requirements.

Start Apache

1) Start the service.

```
$ /usr/local/apache2/bin/apachectl -k start
```

Test the server by navigating to https://<server_FQDN> in a web browser.

NOTE: Make sure to open any host or external firewalls for inbound port 443.

2) Confirm your server is running in FIPS mode. Check the error log for a Notice message that mentions the server is running in FIPS mode. The error log can be retrieved using:

```
$  vi /usr/local/apache2/logs/error_log
```

Appendix A: Supplemental Information

Web Site

Please visit the URLs below for additional information.
NIPRNet: http://iase.disa.mil/pki-pke
SIPRNet: http://iase.disa.smil.mil/pki-pke

Technical Support

Contact technical support at the email address below.
dodpke@mail.mil

Acronyms

AIA	Authority Information Access
CA	Certification Authority
CAC	Common Access Card
CRL	Certificate Revocation List
CSR	Certificate Signing Request
DoD	Department of Defense
DES	Data Encryption Standard
DN	Distinguished Name
FIPS	Federal Information Processing Standards
FQDN	Fully Qualified Domain Name
GDS	Global Directory Service
HTTP	Hyper Text Transfer Protocol
HTTPD	Apache HTTP Server
IP	Internet Protocol
NSS	Network Security Services
OCSP	Online Certificate Status Protocol
PEM	Privacy Enhanced Mail
PKCS	Public Key Cryptography Standard
PKE	Public Key Enablement
PKI	Public Key Infrastructure
SHA	Secure Hash Algorithm
SSL	Secure Socket Layer
STIG	Security Technical Implementation Guide
TLS	Transport Layer Security
UPN	User Principal Name
URL	Uniform Resource Locator

Appendix B: dod-openssl.cfg

NIPR

```
[ req ]
default_bits            = 2048
default_keyfile         = dodserverkey.key
distinguished_name      = req_distinguished_name
default_md              = sha1

dirstring_type = nobmp

[ req_distinguished_name ]
C                         = Country Name (2 letter code)
C_default                 = US
C_min                     = 2
C_max                     = 2

O                         = Organization Name
O_default                 = U.S. Government
O_max                     = 64

0.OU                        = Organizational Unit Name
0.OU_default                = DoD
0.OU_max                    = 64

1.OU                        = Organizational Unit Name
1.OU_default                = PKI
1.OU_max                    = 64

2.OU                        = Organizational Unit Name (CC/S/A, e.g.
USA,USN,DISA,etc.)
2.OU_max                    = 64

CN                        = Common Name (FQDN or routable IP)
CN_max                    = 64
```

SIPR

```
[ req ]
default_bits           = 2048
default_keyfile        = dodserverkey.key
distinguished_name     = req_distinguished_name
default_md             = sha1

dirstring_type = nobmp

[ req_distinguished_name ]
C                        = Country Name (2 letter code)
C_default                = US
C_min                    = 2
C_max                    = 2

O                        = Organization Name
O_default                = U.S. Government
O_max                    = 64

0.OU                       = Organizational Unit Name
0.OU_default               = NSS
0.OU_max                   = 64

1.OU                       = Organizational Unit Name
1.OU_default               = DoD
1.OU_max                   = 64

2.OU                       = Organizational Unit Name (CC/S/A, e.g.
USA,USN,DISA,etc.)
2.OU_max                   = 64

CN                       = Common Name (FQDN or routable IP)
CN_max                   = 64
```

Appendix C: Building OpenSSL and httpd in FIPS mode

The following procedures provide additional guidance for building FIPS-compatible OpenSSL and httpd.

Building FIPS-compatible OpenSSL

After building the OpenSSL FIPS Object Module according to the instructions in http://www.openssl.org/docs/fips/UserGuide-2.0.pdf, a new version of OpenSSL must be built incorporating the Object Module and overwriting any existing version of OpenSSL already installed.

1) Verify and make a note of the install directory of the FIPS Object Module library files, which by default is:

   ```
   /usr/local/ssl/fips-X.X/lib
   ```

2) Determine the install directory of the version of OpenSSL that is currently installed in the operating system by running the following in a terminal:

   ```
   find / -name openssl
   ```

   In the output, look for the lines that end with `/bin/openssl`, `lib/openssl`, and `/include/openssl`. The parent directory listed at the start of these lines will be the install directory to be used when building a new FIPS-compatible version of OpenSSL. For example, if the output lists `/usr/bin/openssl`, `/usr/lib/openssl`, and `/usr/include/openssl`, then the new version of OpenSSL should be installed to `/usr`.

3) Choose a version of OpenSSL that is compatible with the FIPS Object Module according to the FIPS 140-2 Object Module User Guide, and download its source code from http://www.openssl.org/source/.

4) Extract the source code to a temporary directory. Open a terminal into this directory and configure the build with:

   ```
   ./config fips --with-fipslibdir=<install directory of the FIPS object
   module> --prefix=<install directory of openssl>
   ```

   Replace `<install directory of the FIPS object module>` with the directory from Step 1 and `<install directory of openssl>` with the directory from Step 2. In CentOS 5/6, with all default directories, the command would be:

   ```
   ./config fips --with-fipslibdir=/usr/local/ssl/fips-X.X/lib \
   --prefix=/usr
   ```

5) After the `config` command has completed, in the source code directory run:

   ```
   make
   ```

Followed by:

```
make install
```

6) Verify that the newly built OpenSSL has overwritten the existing OpenSSL installation by running the following in a terminal:

   ```
   openssl version
   ```

   The version number displayed should correspond to the version downloaded in Step #3 and contain the word "fips".

Building FIPS-compatible httpd

After FIPS-compatible OpenSSL has been built, httpd needs to be rebuilt so that it links to the new FIPS-OpenSSL module. This build of httpd can coexist with any currently installed version of httpd as long as installed into a separate directory.

1) Download the Unix/Linux source code from http://httpd.apache.org/download.cgi#apache24 .
2) Extract the source code into a temporary directory. Open a terminal into this directory and configure the build with:

   ```
   ./configure --enable-ssl
   ```

   This will enable mod_ssl and install httpd to `/usr/local/apache2` by default. The location can be changed by appending `--prefix=<install directory>` to the command.
3) After the `configure` command has completed, in the source code directory run:

   ```
   make
   ```

   Followed by:

   ```
   make install
   ```

4) Httpd capable of running in FIPS mode using mod_ssl has now been built and installed. Refer to the Reference Manual section of http://httpd.apache.org/docs/2.4/ for further guidance on using this version Apache.

Appendix D: Client Certificate Account Mapping For Custom Applications

This appendix outlines how to enable environment variables to obtain information on the client certificate used to authenticate to Apache. Custom web applications hosted on Apache may have a need to perform account mapping of client certificates to user accounts. Obtaining the information to perform account mapping can be done using mod_ssl or mod_nss environment variables. These environment variables can be used by scripts and custom code to obtain certain certificate attributes, which can be used to map the certificate to a user account.

A backend database is often used to store certificate-to-user account mapping information. For example, the most widely used certificate attribute in the DoD for account mapping is the User Principal Name (UPN). The UPN is contained in the Subject Alternative Name attribute of the DoD CAC email signing certificates. If the UPN will be used to map the certificate to a user account, a table could be created in the database that matches the UPN to the user account. Once the application obtains the UPN from the certificate presented by the client, it could search the database table for a user account that matches the UPN.

Accessing Client Certificate Information with mod_nss

For mod_nss to be able to access the environment variables that contain client certificate information, the nss.conf file must contain the line `NSSOptions +StdEnvVars`. It is recommended that this line is set only where required and is not set for the entire server. For example, if only cgi scripts will use this information and are located in /var/www/cgi-bin, then only set this line for the /var/www/cgi-bin directory using the following syntax in nss.conf:

```
<Directory "/var/www/cgi-bin">
    NSSOptions +StdEnvVars
</Directory>
```

The `NSSOptions +StdEnvVars` line makes available the following environment variables:

Name	Description
SSL_VERSION_INTERFACE	The version of mod_nss the server is running.
SSL_VERSION_LIBRARY	The version of NSS that mod_nss was compiled against.
SSL_PROTOCOL	SSLv2, SSLv3, or TLSv1.
SSL_CIPHER	The cipher the connection is using.
SSL_CIPHER_EXPORT	True if the cipher is an export cipher, false otherwise.

SSL_CIPHER_USEKEYSIZE	Number of bits the cipher is using.
SSL_CIPHER_ALGKEYSIZE	Max number of bits possible in the cipher.
SSL_CLIENT_VERIFY	NONE if no client auth, SUCCESS or FAILED if SSLVerifyCert is set.
SSL_CLIENT_V_START	Client certificate validity start time.
SSL_CLIENT_V_END	Client certificate validity end time.
SSL_CLIENT_V_REMAIN	Number of days that the certificate is valid.
SSL_CLIENT_M_VERSION	X.509 version of the client certificiate.
SSL_CLIENT_M_SERIAL	Serial number of the client certificate.
SSL_CLIENT_A_KEY	Algorithm used for client key.
SSL_CLIENT_A_SIG	Algorithm used for the signature of the client key.
SSL_CLIENT_S_DN	DN of the client certificate.
SSL_CLIENT_S_DN_[C,ST,L,O,OU,CN,T,I,G,S,D,UID,Email]	Components of the client certificate. Only those that exist in the certificate are created.
SSL_CLIENT_I_DN	DN of the client certificate issuer.
SSL_CLIENT_I_DN_[C,ST,L,O,OU,CN,T,I,G,S,D,UID,Email]	Components of the client issuer certificate. Only those that exist in the certificate are created.
SSL_SERVER_DN	DN of the server certificate.
SSL_SERVER_DN_[C,ST,L,O,OU,CN,T,I,G,S,D,UID,Email]	Components of the server certificate. Only those that exist in the certificate are created.
SSL_SERVER_I_DN_[C,ST,L,O,OU,CN,T,I,G,S,D,UID,Email]	Components of the server issuer certificate. Only those that exist in the certificate are created.
SSL_SERVER_M_VERSION	X.509 version of the server certificiate.
SSL_SERVER_M_SERIAL	Serial number of the server certificate.
SSL_SERVER_V_START	Server certificate validity start time.
SSL_SERVER_V_END	Server certificate validity end time.
SSL_SERVER_A_KEY	Algorithm used for server key.
SSL_SERVER_A_SIG	Algorithm used for the signature of the server key.
SSL_SESSION_ID	SSL Session ID.

The UPN is often used for user account mapping in the DoD. However, the variables available when using the above `NSSOptions +StdEnvVars` line in nss.conf will not make the certificate UPN available to applications. To make the UPN available, the nss.conf file must be edited to include the line `NSSOptions +ExportCertData`. This line makes the entire user certificate available to the application, which can parse the certificate for attributes such as the UPN. For performance reasons, this feature should only be made available where needed and not on the entire server.

Adding the `NSSOptions +ExportCertData` line to nss.conf makes the following environment variables available:

Name	Description
SSL_SERVER_CERT	The server certificate in PEM format.
SSL_CLIENT_CERT	The client certificate in PEM format (if available).
SSL_CLIENT_CERT_CHAIN_[0..n]	Each certificate in the client certificate chain in PEM format (including the client certificate itself).

Accessing Client Certificate Information with mod_ssl

For mod_ssl to be able to access the environment variables that contain client certificate information, the ssl.conf file must contain the line `SSLOptions +StdEnvVars`. It is recommended that this line is only set where required and is not set for the entire server. For example, if only cgi scripts will use this information and are located in /var/www/cgi-bin, then only set this line for the /var/www/cgi-bin directory using the following syntax in ssl.conf:

```
<Directory "/var/www/cgi-bin">
    SSLOptions +StdEnvVars
</Directory>
```

The `SSLOptions +StdEnvVars` line makes available the following environment variables:

Variable Name:	Value Type:	Description:
HTTPS	Flag	HTTPS is being used.
SSL_PROTOCOL	String	The SSL protocol version (SSLv2, SSLv3, TLSv1, TLSv1.1, TLSv1.2).
SSL_SESSION_ID	String	The hex-encoded SSL session id.
SSL_CIPHER	String	The cipher specification name.
SSL_CIPHER_EXPORT	String	True if cipher is an export cipher.
SSL_CIPHER_USEKEYSIZE	Number	Number of cipher bits (actually used).
SSL_CIPHER_ALGKEYSIZE	Number	Number of cipher bits (possible).
SSL_COMPRESS_METHOD	String	SSL compression method negotiated.
SSL_VERSION_INTERFACE	String	The mod_ssl program version.
SSL_VERSION_LIBRARY	string	The OpenSSL program version.
SSL_CLIENT_M_VERSION	string	The version of the client certificate.

SSL_CLIENT_M_SERIAL	string	The serial of the client certificate.
SSL_CLIENT_S_DN	string	Subject DN in client's certificate.
SSL_CLIENT_S_DN_x509	string	Component of client's Subject DN.
SSL_CLIENT_I_DN	string	Issuer DN of client's certificate.
SSL_CLIENT_I_DN_x509	string	Component of client's Issuer DN.
SSL_CLIENT_V_START	string	Validity of client's certificate (start time).
SSL_CLIENT_V_END	string	Validity of client's certificate (end time).
SSL_CLIENT_V_REMAIN	string	Number of days until client's certificate expires.
SSL_CLIENT_A_SIG	string	Algorithm used for the signature of client's certificate.
SSL_CLIENT_A_KEY	string	Algorithm used for the public key of client's certificate.
SSL_CLIENT_VERIFY	string	NONE, SUCCESS, GENEROUS or FAILED:reason.
SSL_SERVER_M_VERSION	string	The version of the server certificate.
SSL_SERVER_M_SERIAL	string	The serial of the server certificate.
SSL_SERVER_S_DN	string	Subject DN in server's certificate.
SSL_SERVER_S_DN_x509	string	Component of server's Subject DN.
SSL_SERVER_I_DN	string	Issuer DN of server's certificate.
SSL_SERVER_I_DN_x509	string	Component of server's Issuer DN.
SSL_SERVER_V_START	string	Validity of server's certificate (start time).
SSL_SERVER_V_END	string	Validity of server's certificate (end time).
SSL_SERVER_A_SIG	string	Algorithm used for the signature of server's certificate.
SSL_SERVER_A_KEY	string	Algorithm used for the public key of server's certificate.
SSL_TLS_SNI	string	Contents of the SNI TLS extension (if supplied with ClientHello).

The UPN is often used for user account mapping in the DoD, but the variables available when inserting the `SSLOptions +StdEnvVars` line in ssl.conf will not make the certificate UPN available to applications. In order to make the UPN available, the ssl.conf file must be edited to include the line `SSLOptions +ExportCertData`. This line makes the entire user certificate available to the application, which can parse the certificate for attributes such as the UPN. For performance reasons, this feature should only be made available where needed and not on the entire server.

Adding the `SSLOptions +ExportCertData` line to ssl.conf makes the following environment variables available:

Name	Description
SSL_SERVER_CERT	The server certificate in PEM format.
SSL_CLIENT_CERT	The client certificate in PEM format (if available).
SSL_CLIENT_CERT_CHAIN_[0..n]	Each certificate in the client certificate chain in PEM format (including the client certificate itself).

Appendix E: Configuring SSL Hint List (Acceptable CA Names)

mod_ssl

When a client certificate is requested by mod_ssl, a list of *acceptable Certificate Authority names* -- also known as a *hint list* -- is sent to the client in the SSL handshake. These CA names limit the client certificates that the server will accept to only client certificates issued by CAs in the *hint list*. Most browsers will use this *hint list* to filter the presented client certificate options to only client certificates that are issued by one of the acceptable CAs. This feature is helpful in helping the user choose the correct authenticating certificate when they have multiple options in their certificate store.

Apache with mod_ssl can be configured to provide a customized *hint list* during SSL handshakes. The directive that can be used to configure this list is **SSLCADNRequestFile.**

If no *hint list* is configured, then the set of acceptable CA names sent to the client is the names of all the CA certificates given by the **SSLCACertificateFile** directive, i.e., the names of the CAs which will actually be used to verify the client certificate.

Using the **SSLCADNRequestFile** directive:

1) Add the **SSLCADNRequestFile** directive into the SSL Global Context body of the file. **SSLCADNRequestFile** must specify an all-in-one file containing a concatenation of PEM-encoded CA certificates.
2) To create a DNRequestFile with a sub-set of the DoD certificates, copy the desired certificates from the “alldodcerts.pem” file that was created in the **Trusted Certificate Configuration** section and paste them into a new file named `ca-names.pem`.

   Example:

   ```
   SSLCADNRequestFile /usr/local/apache2/conf/ca-names.pem
   ```
3) Restart the server.

mod_nss

Currently we have not identified a way to customize the hint list in mod_nss other than altering the CA certificates that the serer trusts. In mod_nss, the CA certificates that are marked as Root certificates in the NSS database are included in the hint list.

Appendix F: References

[1] Bob Lord, "Red Hat Speaks", Red Hat Magazine, December 2004 (http://www.redhat.com/magazine/002dec04/departments/red_hat_speaks/).

[2] Open Source Initiative, http://www.opensource.org/ (2011).

[3] OpenSSL (2011), Cryptography and SSL/TLS Toolkit, http://www.openssl.org/ (2011).

[4] "SSL/TLS Strong Encryption: How-To. Apache HTTP Server Version 2.0", The Apache Foundation, http://httpd.apache.org/docs/2.0/ssl/ssl_howto.html (2011).

[5] "mod_nss", Fedora Project, http://directory.fedoraproject.org/wiki/Mod_nss (2011).

[6] Red Hat Inc and Sun Microsystems Inc, 18 July 2007, "NSS Cryptographic Module Version 3.11.4 FIPS 140-2 Non-Proprietary Security Policy Level 1 and Validation", http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140sp/140sp814.pdf (2011).

[7] "Apache Module mod_ssl. Apache HTTP Server Version 2.0", The Apache Foundation, http://httpd.apache.org/docs/2.0/mod/mod_ssl.html (2011).