Linux on z Systems

Configuring an Apache mod_nss server to exploit z Systems cryptographic hardware

Linux on z Systems

Configuring an Apache mod_nss server to exploit z Systems cryptographic hardware

Before using this information and the product it supports, read the information in “Notices” on page 25.

Edition notices

Contents

About this publication

This white paper provides information about how to configure an Apache HTTPS server with mod_nss to exploit the cryptographic hardware functions available with IBM® z Systems™ cryptographic hardware. The scenario provided in this white paper uses Red Hat Enterprise Linux (RHEL) 7. The scenario was tested using *Red Hat Enterprise Linux (RHEL) 7.1*.

Authors

Crypto for Linux on z Systems team:
Patrick Steuer, Dr. Reinhard Buendgen, George C. Wilson

Chapter 1. Introduction

This white paper describes how to configure an Apache HTTPS server with mod_nss under RHEL 7 to exploit the cryptographic hardware functions available with IBM z Systems. *A scenario is provided that you might wish to repeat in your own installation*.

There are two security modules that can be used for setting up the SSL/TLS implementation in an Apache HTTPS server:

- mod_ssl
- mod_nss

The scenario uses the *mod_nss* security module.

In addition, the scenario was tested using:

- A RHEL7.1 operating system that was installed in an LPAR of an IBM zEnterprise® EC12 System.
- Central Processor Assist for Cryptographic Functions (CPACF).
- A Crypto Express4 (CEX4) adapter.

The tools and path/file names used in this white paper might be different for other Linux software distributions.

The appendix provides a sample SELinux policy module that you can use as a template.

Chapter 2. Prerequisite tasks

This topic describes the prerequisite tasks that must be completed before configuring the Apache HTTPS server under RHEL 7 to exploit the cryptography hardware that is provided by z Systems.

The scenario provided in this white paper uses this cryptographic hardware:

- CPACF feature (feature code LIC3863) supporting 3DES, AES, and SHA operations.
- (Optionally) A Crypto Express (CEX) adapter in either accelerator or CCA co-processor mode to support RSA operations.

For a description of the hardware and software used for testing the scenario, see .

In the steps described here, the information that you must enter is shown in **bold** font.

Installing the required packages

To repeat the scenario, you must install various packages.

To start the installation of the required packages, enter:

```
# yum install libica opencryptoki httpd mod_nss
```

To avoid any possible conflicts, remove the module mod_ssl (if installed on your system):

```
# yum erase mod_ssl
```

Store an html file containing your Web page in `/var/www/httpd/`. This Web page will be displayed when a connection is made to the Apache HTTPS server.

Loading the zcrypt device driver

To access Crypto Express (CEX) adapters, the zcrypt device driver must be loaded into the kernel.

The steps described in this topic only apply if your configuration includes a Crypto Express adapter in accelerator or CCA coprocessor mode. If your configuration does *not* include a Crypto Express adapter in accelerator or CCA coprocessor mode, all RSA operations will be performed by the software.

1. Load the zcrypt device driver:

   ```
   # modprobe ap
   ```

2. Specify that the zcrypt device driver should be automatically loaded on boot by creating a script called **ap.modules** in `/etc/sysconfig/modules`:

```
#!/usr/bin/bash
modprobe ap
```

3. Set the appropriate permissions:

```
# chmod 770
/etc/sysconfig/modules/ap.modules
```

```
# chown root:root
/etc/sysconfig/modules/ap.modules
```

For further details about the zcrypt device driver, see *Device Drivers, Features, and Commands*, SC33-8411.

Configuring the openCryptoki ica token

openCryptoki is a PKCS#11 implementation. Using *tokens* and *slots*, the PKCS#11 standard unifies the way in which applications access cryptographic objects. The openCryptoki *ica token* is used to perform *clear-key* cryptography (where the key exists somewhere in the software stack) by exploiting z Systems hardware.

The ica token provides access to the cryptographic hardware of z Systems in order to perform cryptographic operations. It exploits z Systems CPACF cryptographic hardware in the CPU (using 3DES/AES/SHA) and in the CEX adapters (using RSA).

To configure the openCryptoki ica token:

1. Add to the pkcs11 group, the users who should be allowed to access the openCryptoki library:

```
# usermod -aG pkcs11 root
# usermod -aG pkcs11 apache
```

2. Enable the pkcsslot daemon (which manages access to the security tokens):

```
# systemctl enable pkcsslotd.service
```

3. Start the slot daemon:

```
# systemctl start pkcsslotd.service
```

4. Use the **pkcsconf** command to locate the slot number (**slot 1** in this scenario) of OpenCryptoki's *ica token*. In the default configuration listing that is shown below, the ICA token is Token # **1**. Your listing will probably differ from this listing.

```
# pkcsconf -t
Info:Token #1 Info:
 Label:  IBM ICA  PKCS #11
    Manufacturer: IBM Corp.
    Model: IBM ICA
    Serial Number: 123
 Flags: 0x880045 (RNG|LOGIN_REQUIRED|CLOCK_ON_TOKEN|USER_PIN_TO_BE_CHANGED|
     SO_PIN_TO_BE_CHANGED)
 Sessions: 0/-2
 R/W Sessions: -1/-2
 PIN Length: 4-8
 Public Memory: 0xFFFFFFFF/0xFFFFFFFF
 Private Memory: 0xFFFFFFFF/0xFFFFFFFF
 Hardware Version: 1.0
 Firmware Version: 1.0
 Time: 14:16:45
...
```

5. Set a token label (**myicatoken** below) using the **-I** option. The pkcsconf tool is used to initialize the ica token in slot 1 (**-c 1** below). The default SO ("slot operator" or "security officer") pin is 87654321.

```
# pkcsconf -I -c 1
Enter the SO PIN: 87654321
Enter a unique token label: myicatoken
```

6. Change the default SO pin using the **-P** option. In the example below, the new pin is 13243546.

```
# pkcsconf -P -c 1
Enter the SO PIN: 87654321
Enter the new SO PIN: 13243546
Re-enter the new SO PIN: 13243546
```

7. Initialize the user pin using the **-u** option. In the example below, the new user pin is 25345867.

```
# pkcsconf -u -c 1
Enter the SO PIN: 13243546
Enter the new user PIN: 25345867
Re-enter the new user PIN: 25345867
```

Note: To ensure the SO has no access to the token, you should change the user pin as soon as a user is granted access. To do so, enter:

```
# pkcsconf -p -c 1
```

Disabling the firewall and SELinux

Temporarily disable both the firewall and SELinux. This is required before you can reboot the Linux system and test if the prerequisite steps were successfully implemented.

1. Disable the firewall on boot:

```
# systemctl disable firewalld.service
```

2. Disable SELinux by editing file `/etc/selinux/config` and changing the line:
   `SELINUX=[...]`

   to
   `SELINUX=disabled`

Checking that the prerequisite tasks were successfully completed

Various checks are required before you can start to configure the Apache HTTPS server under RHEL 7 to exploit z Systems cryptographic hardware functions.

Note: Steps 2 and 3 of this topic are only required if you are using a Crypto Express adapter (for details, see "Installing the required packages" on page 3).

1. Reboot the RHEL 7 operating system:

```
# reboot
```

2. Check that the crypto device driver has been loaded:

```
# lsmod | grep zcrypt
zcrypt_msgtype6        18217  1
zcrypt_cex4            12819  2
zcrypt_api             30536  2 [...]
ap                     35201  3 [...]
```

   Note that the output shown above might differ if you are using a Crypto Express (CEX) adapter in accelerator mode.

3. Check that the required cryptographic adapters are online (their device-IDs and firmware loads are shown below as underlined).

```
# lszcrypt -V
card00: CEX4C       online
card01: CEX4C       online
card03: CEX4P       online
```

   At least one online adaper of type Accelerator (A) or CCA-Coprocessor (C) is required. In the above example, we see that two coprocessors have been made available.

   You can set a cryptographic adapter online using command **chzcrypt -e <device id>**.

   You can set a cryptographic adapter offline using command **chzcrypt -d <device id>**.

4. Check that SELinux is disabled:

```
# sestatus
SELinux status:  disabled
```

5. Check that the firewall is disabled:

```
# systemctl status firewalld.service
firewalld.service - firewalld - dynamic firewall daemon
Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled)
Active: inactive (dead)
```

6. Check that the slot daemon is running:

```
# systemctl status pkcsslotd.service
pkcsslotd.service - Daemon which manages cryptographic hardware tokens for the
                    openCryptoki package
   Loaded: loaded (/usr/lib/systemd/system/pkcsslotd.service; enabled)
   Active: active (running) since Wed 2015-08-05 17:19:41 CEST; 3min 59s ago
  Process: 682 ExecStart=/usr/sbin/pkcsslotd (code=exited, status=0/SUCCESS)
 Main PID: 694 (pkcsslotd)
   CGroup: /system.slice/pkcsslotd.service
       └─694 /usr/sbin/pkcsslotd
```

7. Check that the ica token is initialized:

```
# pkcsconf -t -c1
Token #1 Info:
 Label: myicatoken
 Manufacturer: IBM Corp.
 Model: IBM ICA
 Serial Number: 123
 Flags: 0x44D (RNG|LOGIN_REQUIRED|USER_PIN_INITIALIZED|CLOCK_ON_TOKEN|TOKEN_INITIALIZED)
 Sessions: 0/-2
 R/W Sessions: -1/-2
 PIN Length: 4-8
 Public Memory: 0xFFFFFFFF/0xFFFFFFFF
 Private Memory: 0xFFFFFFFF/0xFFFFFFFF
 Hardware Version: 1.0
 Firmware Version: 1.0
 Time: 17:24:23
```

8. Check that users `root` and `apache` can communicate with the slot daemon:

```
# groups root apache
root : root pkcs11
apache : apache pkcs11
```

For further details about using openCryptoki's pkcsslot daemon, see *libica Programmer's Reference*, SC34-2602.

Chapter 3. Creating an nss certificate database

You must now create a new nss certificate database for use with the Apache HTTPS server and z Systems cryptographic hardware.

In the steps described here, the information that you must enter is shown in **bold** font.

Creating a directory for the nss database

A directory must be created for use with the nss database. Later, the required certificates will be stored in this directory.

1. Create the directory:

```
mkdir /etc/httpd/nss
```

2. Initialize the new database using the option **-N** (New) option. In this scenario, the new database password used for initializing the database is 19283746.

```
# certutil -N -d /etc/httpd/nss
Enter a password which will be used to encrypt your keys.
The password should be at least 8 characters long,
and should contain at least one non-alphabetic character.

Enter new password: 19283746
Re-enter password: 19283746
```

Adding the openCryptoki module to the nss database

Add the openCryptoki module to the nss database and specify it as the default provider. PKCS11 modules in the database are managed by the **modutil** tool.

1. Insert a statement `description=<your_slot_name>` immediately below `stdll=libpkcs11_ica.so` in file `/etc/opencryptoki.conf`. In the example shown below, the name `ICA` is used.

```
slot1
{
stdll=libpkcs11_ica.so
description=ICA
}
```

2. Add the openCryptoki module (in this scenario, the name used is `opencryptoki`) to the nss database:

```
# modutil -dbdir /etc/httpd/nss -add opencryptoki -libfile
      /usr/lib64/opencryptoki/PKCS11_API.so
WARNING: Performing this operation while the browser is running could cause
corruption of your security databases. If the browser is currently running,
you should exit browser before continuing this operation. Type
'q <enter>' to abort, or <enter> to continue:

Module "opencryptoki" added to database.
```

3. Disable all slots of the openCryptoki module and then enable only the ICA slot:

```
# modutil -dbdir /etc/httpd/nss -disable opencryptoki
# modutil -dbdir /etc/httpd/nss -enable opencryptoki -slot ICA
```

4. Set OpenCryptoki to be the default provider for the required cryptographic mechanisms:

```
modutil -dbdir /etc/httpd/nss -default opencryptoki -mechanisms RSA:AES:DES:RANDOM
WARNING: Performing this operation while the browser is running could cause
corruption of your security databases. If the browser is currently running,
you should exit browser before continuing this operation. Type
'q <enter>' to abort, or <enter> to continue:

Successfully changed defaults.
```

5. List the available cryptographic modules by entering:

```
# modutil -dbdir /etc/httpd/nss -list
Listing of PKCS #11 Modules
-----------------------------------------------------------
  1. NSS Internal PKCS #11 Module
  slots: 2 slots attached
 status: loaded

  slot: NSS Internal Cryptographic Services
 token: NSS Generic Crypto Services

  slot: NSS User Private Key and Certificate Services
 token: NSS Certificate DB

  2. opencryptoki
 library name: /usr/lib64/opencryptoki/PKCS11_API.so
  slots: 2 slots attached
 status: loaded

  slot: ICA
 token: myicatoken

  slot: Linux
 token: IBM OS PKCS#11
-----------------------------------------------------------
```

6. Display detailed information about the openCryptoki module by entering:

```
# modutil -dbdir /etc/httpd/nss -list opencryptoki
-----------------------------------------------------------
Name: opencryptoki
Library file: /usr/lib64/opencryptoki/PKCS11_API.so
Manufacturer: IBM
Description: Meta PKCS11 LIBRARY
PKCS #11 Version 2.20
Library Version: 3.2
Cipher Enable Flags: None
Default Mechanism Flags: RSA:DES:AES
  Slot: ICA
  Slot Mechanism Flags: RSA:DES:AES
  Manufacturer: IBM
  Type: Software
  Version Number: 0.0
  Firmware Version: 0.0
  Status: Enabled
  Token Name: myicatoken
  Token Manufacturer: IBM Corp.
  Token Model: IBM ICA
  Token Serial Number: 123
  Token Version: 1.0
  Token Firmware Version: 1.0
  Access: NOT Write Protected
  Login Type: Login required
  User Pin: Initialized

  Slot: Linux
  Slot Mechanism Flags: RSA:DES:AES
  Manufacturer: IBM
  Type: Software
  Version Number: 0.0
  Firmware Version: 0.0
  Status: DISABLED (user disabled)
  Token Name: IBM OS PKCS#11
  Token Manufacturer: IBM Corp.
  Token Model: IBM SoftTok
  Token Serial Number: 123
  Token Version: 1.0
  Token Firmware Version: 1.0
  Access: NOT Write Protected
  Login Type: Login required
  User Pin: NOT Initialized
-----------------------------------------------------------------
```

Create a self-signed CA server certificate

In this topic, we create a server certificate that will be used for configuring the https server.

A server certificate is a data object that is used to associate a server with its public key. Certificates are digitally signed by a Certifiate Authority (CA) to prevent them from being manipulated.

1. In a production environment, the server certificate request would be signed by a recognized third-party (that is, a CA). However, for testing purposes our own self-signed CA server certificate is created (with the options "-S -x") called `testca`. The private key associated with `testca` will later be used to sign the server certificate.
   - In example below, the validity in months is specified with options **-v 48** together with a unique serial number **-m 5555**.
   - Certificates hold trust attributes in three different categories: **ssl**, **email**, and **object signing**.
   - The trust attributes for each category can be set using the (trust) options **-t <ssl>,<email>,<object signing>**.

- The CA in this example has trust settings **CTu**,**CTu**,**CTu** (where **C** means "trusted CA for client authentication", **T** means "trusted CA", and **u** means "user").
- Certificate extensions indicate how a certificate should be used. If an application does not recognize the extensions marked as critical, the certificate must not be accepted. Non-critical extensions may be ignored if they are not recognized but must be processed if recognized. In this example, you specify the options **-1** (keyUsage), **-2** (basic constraint extension) and **-5** (nsCertType). You are prompted to select the appropriate extensions for our CA, as shown in the example below.

```
# certutil -S -d /etc/httpd/nss -n testca -s "CN=Certificate Shack, O=example.com,
      C=US" -x -t CTu,CTu,CTu -g 2048 -m 5555 -v 48 -h myicatoken -1 -2 -5
Enter Password or Pin for "myicatoken": 25345867
...

Continue typing until the progress meter is full:

|************************************************************|

Finished.  Press enter to continue:


Generating key.  This may take a few moments...
```

2. Select option **5** (**Cert signing key**) as shown below:

```
  0 - Digital Signature
  1 - Non-repudiation
  2 - Key encipherment
  3 - Data encipherment
  4 - Key agreement
  5 - Cert signing key
  6 - CRL signing key
  Other to finish
 > 5
```

3. Select option **9** (**Other to finish**) as shown below:

```
  0 - Digital Signature
  1 - Non-repudiation
  2 - Key encipherment
  3 - Data encipherment
  4 - Key agreement
  5 - Cert signing key
  6 - CRL signing key
  Other to finish
 > 9
```

4. Select the options **n**, **y**, and **y** as shown below:

```
Is this a critical extension [y/N]?
n
Is this a CA certificate [y/N]?
y
Enter the path length constraint, enter to skip [<0 for unlimited path]: > 10
Is this a critical extension [y/N]?
y
```

5. Select option **5** (**SSL CA**) as shown below:

```
 0 - SSL Client
 1 - SSL Server
 2 - S/MIME
 3 - Object Signing
 4 - Reserved for future use
 5 - SSL CA
 6 - S/MIME CA
 7 - Object Signing CA
 Other to finish
> 5
```

6. Select option **6** (**S/MIME CA**) as shown below:

```
 0 - SSL Client
 1 - SSL Server
 2 - S/MIME
 3 - Object Signing
 4 - Reserved for future use
 5 - SSL CA
 6 - S/MIME CA
 7 - Object Signing CA
 Other to finish
> 6
```

7. Select option **7** (**Object Signing CA**) as shown below:

```
 0 - SSL Client
 1 - SSL Server
 2 - S/MIME
 3 - Object Signing
 4 - Reserved for future use
 5 - SSL CA
 6 - S/MIME CA
 7 - Object Signing CA
 Other to finish
> 7
```

8. Select option **9** (**Other to finish**) as shown below:

```
 0 - SSL Client
 1 - SSL Server
 2 - S/MIME
 3 - Object Signing
 4 - Reserved for future use
 5 - SSL CA
 6 - S/MIME CA
 7 - Object Signing CA
 Other to finish
> 9
```

9. Select option **n** as shown below:

```
Is this a critical extension [y/N]
n
Notice: Trust flag u is set automatically if the private key is present.
```

Create a certificate request file

A *certificate request file* specifies the attributes of the certificate that is to be requested.

Using the **certutil** command with **-R** option (request file), you can now create a certificate request file (`tmpcertreq`) for the https server certificate in the token.

- The path of the nss database is specified using the **-d** option.

- The subject name is specified using the **-s** option.
- Replace **<FQDN>** with your fully-qualified domain name (which you can obtain using the **hostname -f** command).
- The name and location of the output file is specified using the **-o** option.
- The certificate will contain a 2048-bit RSA key pair that is generated using the **-g** option.
- The certificate request file for the server certificate is created using the **-h** option together with the token name "myicatoken" (token name "myicatoken" was previously used to initialize the token).
- The requested password is the user PIN of the ICA token that was set in Step 7 of topic "Configuring the openCryptoki ica token" on page 4.

```
# certutil -R -d /etc/httpd/nss -s "CN=<FQDN>, O=example.com, C=US" -o
      /etc/httpd/nss/tmpcertreq -g 2048 -h myicatoken
Enter Password or Pin for "myicatoken": 25345867
...

Continue typing until the progress meter is full:

|************************************************************|

Finished.  Press enter to continue:


Generating key.  This may take a few moments...
```

Create a server certificate issued by own Certificate Authority

Create a server certificate that will be issued by the Certificate Authority (CA) that you control.

1. In the example below, you use the **-C** (certificate) request option. The **-i** option specifies the name and path of the request file (in this scenario, "tmpcertreq"). You create the server certificate issued by our CA in our token using options **-c myicatoken:testca**.

   In this scenario, "myicatoken" is the name used for the ica token, and "testca" is the name used for the self-signed CA server certificate. The password entered here is the user PIN of the ICA token (which was set in Step 7 of topic "Configuring the openCryptoki ica token" on page 4). For our server certificate **tmpcert.der** enter:

   ```
   # certutil -C -d /etc/httpd/nss -c myicatoken:testca -i /etc/httpd/nss/tmpcertreq -o
         /etc/httpd/nss/tmpcert.der -m 5556 -v 48 -1 -5
   Enter Password or Pin for "myicatoken": 25345867
   ```

2. Select option **2** (**Key encipherment**) as shown below:

   ```
     0 - Digital Signature
     1 - Non-repudiation
     2 - Key encipherment
     3 - Data encipherment
     4 - Key agreement
     5 - Cert signing key
     6 - CRL signing key
     Other to finish
    > 2
   ```

3. Select option **9** (**Other to finish**) as shown below:

```
  0 - Digital Signature
  1 - Non-repudiation
  2 - Key encipherment
  3 - Data encipherment
  4 - Key agreement
  5 - Cert signing key
  6 - CRL signing key
  Other to finish
 > 9
```

4. Select option **n** as shown below:

```
Is this a critical extension [y/N]?
n
```

5. Select option **1** (**SSL Server**) as shown below:

```
  0 - SSL Client
  1 - SSL Server
  2 - S/MIME
  3 - Object Signing
  4 - Reserved for future use
  5 - SSL CA
  6 - S/MIME CA
  7 - Object Signing CA
  Other to finish
 > 1
```

6. Select option **9** (**Other to finish**) as shown below:

```
  0 - SSL Client
  1 - SSL Server
  2 - S/MIME
  3 - Object Signing
  4 - Reserved for future use
  5 - SSL CA
  6 - S/MIME CA
  7 - Object Signing CA
  Other to finish
 > 9
```

7. Select option **n** as shown below:

```
Is this a critical extension [y/N]?
n
```

On completion of the above steps, the self-signed certificate `testca` is now stored in the nss database.

Import a server certificate into the ica token

Import the server certificate into the openCryptoki ica token.

1. Import the server certificate into our token called **testcert** using option **-A** (add):

```
# certutil -A -d /etc/httpd/nss -n testcert -t u,u,u -i /etc/httpd/nss/tmpcert.der
      -h myicatoken
Notice: Trust flag u is set automatically if the private key is present.
```

2. Display the contents of the token:

```
# certutil -K -d /etc/httpd/nss -h myicatoken
certutil: Checking token "myicatoken" in slot "Linux"
Enter Password or Pin for "myicatoken": 25345867
< 0> rsa      49fe255b05e746c08b4a11d7ac5e5c9f68d8fe5a   myicatoken:testcert
< 1> rsa      f11fa89f1d630d188b7da752c1c004f48b774235   myicatoken:testca
```

   The password that you enter above was set in Step 7 of topic "Configuring the openCryptoki ica token" on page 4.

3. Create a file `password.txt` in `/etc/httpd/nss` which contains the passwords for the database and token. This file should look the following:

```
internal:19283746
myicatoken:25345867
```

4. Set reasonable permissions for the database and password file:

```
# chmod 640 /etc/httpd/nss/*.db
# chown root:apache /etc/httpd/nss/*.db
# chmod 640 /etc/httpd/nss/password.txt
# chown root:apache /etc/httpd/nss/password.txt
```

Chapter 4. Configure and start the Apache HTTPS server

You customize the Apache HTTPS server by tailoring the entries contained in file `/etc/httpd/conf.d/nss.conf`.

In the steps described here, the information that you must enter is shown in **bold** font.

1. To ensure all TCP/IP ports use secure communication, change the following two lines:

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

2. Enable or disable cipher suites by selecting either **+** (enable) or **-** (disable) next to the cipher suites for which hardware support is available on your system. For example:

   ```
   NSSCipherSuite +rsa_aes_256_sha,+rsa_aes_128_sha,+rsa_3des_sha
   ```

3. Specify the server protocols you wish to use. For example, one of:

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

4. Change the following lines to include the server certificate stored in the nss database (as described in Chapter 3, "Creating an nss certificate database," on page 9):

   ```
   NSSCertificateDatabase /etc/httpd/nss
   NSSNickname myicatoken:testcert
   ```

5. Specify the passphrases that were created in the password file:

   ```
   NSSPassPhraseDialog  file:/etc/httpd/nss/password.txt
   ```

6. Finally, specify that the Apache HTTPS server will be started when Linux is started. In addition, start the Apache HTTPS server immediately.

   ```
   # systemctl enable httpd.service
   # systemctl start httpd.service
   ```

Chapter 5. Verifying that cryptographic operations work correctly

You must verify that cryptographic operations can be successfully performed on IBM z Systems cryptographic hardware. To do so, insert the file `index.html` (that you previously created) into the directory `/var/www/html/`.

You can now access your Web site using any supported Web browser. To display the count of cryptographic operations that were performed, enter:

```
# icastats -A
```

To reset the counters used for counting the number of cryptographic operations that were performed, enter:

```
# icastats -R
```

After resetting the counters, access the Web page that you wish to secure via openCryptoki and check that the counters that monitor hardware cryptographic operations have increased.

To investigate any errors that might have occurred during cryptographic operations, enter:

```
# cat /var/log/httpd/error_log
```

To debug errors that might have occurred during cryptographic operations, enter:

```
# openssl s_client -connect <FQDN>:443 -<protocol> -debug
```

(replacing <FQDN> and <protocol> with your own values).

Chapter 6. Completing the configuration

There are various tasks that you must complete before you can use the Apache HTTPS server for cryptographic operations in a production environment.

In the steps described here, the information that you must enter is shown in **bold** font.

Enabling the firewall

Enable your firewall so that it can work together with the Apache HTTPS server.

1. Start the firewall, and then give the http and https services permission to access the firewall.

```
# systemctl start firewalld.service
# firewall-cmd --add-service http --permanent
success
# firewall-cmd --add-service https --permanent
success
```

2. Restart the firewall in order to make your permission changes active and automatically enable the firewall service on boot.

```
# systemctl restart firewalld.service
# systemctl enable firewalld.service
ln -s '/usr/lib/systemd/system/firewalld.service'
  '/etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service'
ln -s '/usr/lib/systemd/system/firewalld.service'
  '/etc/systemd/system/basic.target.wants/firewalld.service'
```

Enabling SELinux

Enable SELinux so that it can work together with the Apache HTTPS server.

1. Obtain the tools used in this topic by running this command:

```
# yum install policycoreutils-python selinux-policy-devel
```

2. Set the following SELinux booleans to value 1:

```
# setsebool -P httpd_unified 1
# setsebool -P daemons_enable_cluster_mode 1
# setsebool -P httpd_run_stickshift 1
```

3. Set type `pkcsslotd_lock_t` in the security context of `/var/lock/opencryptoki` and all folders and sub-directories:

```
# chcon -R -t pkcsslotd_lock_t /var/lock/opencryptoki/
```

4. Write an SELinux policy module with the name `httpd-plus.te`. The source code for policy module `httpd-plus.te` is provided in "Appendix. SELinux policy module" on page 23. To build and install policy module `httpd-plus.te`, enter:

```
# make -f /usr/share/selinux/devel/Makefile
# semodule -i httpd-plus.pp
```

5. Restart the slot daemon and Apache HTTPS server:

```
# systemctl restart pkcsslotd.service
# systemctl restart httpd.service
```

Checking that the Apache HTTPS server and SELinux work together

Check that the Apache HTTPS server and SELinux policy module work together correctly.

1. Check that the policy module `httpd-plus.te` has been correctly installed:

```
# semodule -l | grep httpd-plus
# httpd-plus 1.0.3
```

2. Repeat the checks described in Chapter 5, "Verifying that cryptographic operations work correctly," on page 19 to confirm that the Apache HTTPS server is running and correctly using the z Systems cryptographic hardware.

You have now successfully completed the topics and steps described in this white paper! You should now have an Apache HTTPS server with mod_nss under RHEL 7 that exploits the cryptographic hardware functions available with IBM z Systems.

Appendix. SELinux policy module

This appendix provides the source code for the SELinux policy module `httpd-ock.te`.

```
policy_module(httpd-ock, 1.0.3)

#
#  Additional permissions between httpd and pkcsslotd to allow
#  mod_nss to work with openCryptoki.
#

gen_require(`
        type httpd_t;
        type pkcs_slotd_t;
        type pkcs_slotd_exec_t;
')


# Allow httpd access to pkcsslotd objects

allow httpd_t pkcs_slotd_exec_t:file getattr;         # Allow httpd to access the pkcsslotd binary

pkcs_admin_slotd(httpd_t,system_r)                    # Allow httpd to administer pkcsslotd

allow httpd_t pkcs_slotd_t:shm rw_shm_perms;          # Allow httpd to access pkcsslotd's shared
                                                      # memory segement


# Allow pkcsslotd access to httpd objects

apache_signal(pkcs_slotd_t)                           # Allow pkcsslotd to send signals except
                                                      # signull to httpd

apache_signull(pkcs_slotd_t)                          # Allow pkcsslotd to send signull to httpd
                                                      # to check if it is available

allow pkcs_slotd_t httpd_t:dir search_dir_perms;      # Allow pkcsslotd to search httpd directories

allow pkcs_slotd_t httpd_t:file read_file_perms;      # Allow pkcsslotd to read httpd files
```

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing 2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating

platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information in softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

Adobe is either registered trademarks or trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Terms and conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

Personal Use: You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these publications, or any portion thereof, without the express consent of the manufacturer.

Commercial Use: You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of the manufacturer.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any data, software or other intellectual property contained therein.

The manufacturer reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by the manufacturer, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

THE MANUFACTURER MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THESE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Printed in USA