Apache HTTP Server

nShield® HSM Integration Guide - CHIL

Table of Contents

Chapter 1. Introduction

The Apache HTTP Server 2.4.6 integrates with the Entrust nShield® Hardware Security Module (HSM) to provide a secure web server solution. The nShield HSMs are hardened, tamper-resistant cards which perform encryption, digital signing and key generation on behalf of an extensive range of commercial and custom-built applications, including certificate authorities, and code signing.

The benefits of using an nShield Hardware Security Module (HSM) with the Apache HTTP Server include:

- Secure storage of the private key.
- FIPS 140 Level 3 validated hardware.
- Improved server performance by offloading the cryptographic processing.
- Full life cycle management of the keys.
- Failover support.
- Load balancing between HSMs.

Throughout this guide, the term HSM refers to nShield Solo and nShield Connect units. (nShield Solo products were formerly known as nShield).

This guide describes how to use the nShield Cryptographic Hardware Interface Library (CHIL) interface to integrate the HSM and Apache HTTP Server.

1.1. Product configurations

We have successfully tested nShield HSM integration with the server in the following configurations:

Operating System	Apache version	OpenSSL version	Security World Software version	nShield Solo support	nShield Connect support
Red Hat Enterprise Linux 7 x 64-bit	2.4.6	1.0.2k-fips	12.60.3 *	Yes	Yes

* The nShield 12.40 Compatibility Package is required for the Cryptographic Hardware Interface Library (CHIL) plugin. To obtain the package, contact contact

Entrust nShield Support, https://nshieldsupport.entrust.com.

Access to the Entrust nShield Support Portal is available to customers under maintenance. To request an account, contact nshield.support@entrust.com.

1.2. Supported nShield functionality

Feature	Support	Feature	Support	Feature	Support
Key Generation	Yes	1-of-N Operator Card Set	Yes	FIPS 140 Level 3 Support	Yes
Key Management	Yes	K-of-N Operator Card Set	Yes	Load Sharing	Yes
Key Import	Yes	Softcards	Yes	Fail Over	Yes
Key Recovery	Yes	Module-only Key	Yes		

1.3. Requirements

Ensure that you have supported versions of the nShield, Apache, and third-party products. See Product configurations.

Consult the security team in your organization for a suitable setting of the SE Linux policy to allow the web server read access to the files in `/opt/nfast`.

To perform the integration tasks, you must have:

- `root` access on the operating system.
- Access to `nfast` and `httpd` accounts.

Before starting the integration process, familiarize yourself with:

- The documentation for the HSM.
- The documentation and setup process for the Apache HTTP server.

Before using the nShield software, you need to know:

- The number and quorum of Administrator Cards in the Administrator Card Set (ACS), and the policy for managing these cards.

- Whether the application keys are protected by the module or an Operator Card Set (OCS) with or without a pass phrase.
- The number and quorum of Operator Cards in the OCS, and the policy for managing these cards.
- Whether the security world should be compliant with FIPS 140 Level 3.

Entrust recommends that you allow only unprivileged connections unless you are performing administrative tasks.

For more information, refer to the *User Guide* and *Installation Guide* for the HSM.

1.4. More information

For more information about OS support, contact your Apache HTTP Server sales representative or Entrust nShield Support, https://nshieldsupport.entrust.com.

Access to the Entrust nShield Support Portal is available to customers under maintenance. To request an account, contact nshield.support@entrust.com.

Chapter 2. Procedures

Integration procedures include:

- Installing the HSM.
- Installing the Security World Software and create the security world.
- Installing the Apache HTTP Server.
- Testing CHIL.
- Configuring the Apache HTTP Server to use the HSM.

This chapter describes these procedures.

2.1. Installing the HSM

Install the HSM by following the instructions in the *Installation Guide* for the HSM.

We recommend that you install the HSM before configuring the Security World Software with your Apache HTTP Server.

2.2. Installing the Security World Software and creating the security world

To install the Security World Software and create the security world:

1. On the computer that you want to make the Apache HTTP Server, install the latest version of the Security World Software as described in the *Installation Guide* for the HSM.

   

   We recommend that you uninstall any existing nShield software before installing the new nShield software.

2. Create the security world as described in the *User Guide*, creating the ACS and OCS that you require.

2.3. Installing and configuring the Apache HTTP Server

To install the Apache HTTP Server:

```
sudo yum install httpd-tools openssl-libs mod_ssl
```

2.4. Testing CHIL

The nShield 12.40 Compatibility Package is required for the Cryptographic Hardware Interface Library (CHIL) plugin. Because this version of the library needs a gen2 Security World, either an old world needs to be loaded, or the utility `new-world-1240` needs to be used to create a suitable Security World.

To check that CHIL is working:

```
# export LD_LIBRARY_PATH=/opt/nfast/toolkits/hwcrhk/
# openssl engine -t chil
(chil) CHIL hardware engine support
[ available ]
```

2.5. Configuring the Apache HTTP Server to use the HSM

2.5.1. Environment settings

For convenience:

```
export PATH=$PATH:/opt/nfast/bin
```

In `/etc/sysconfig/httpd` add the line

```
LD_LIBRARY_PATH=/opt/nfast/toolkits/hwcrhk
```

2.5.2. Set up Apache to use the CHIL library

Generate an embed key. Ensure that the key files are output to your `home` directory or another working directory.

```
# generatekey embed
protect: Protected by? (token, module) [token] > module
size: Key size? (bits, minimum 1024) [2048] >
OPTIONAL: pubexp: Public exponent for RSA key (hex)? []
> embedsavefile: Filename to write key to? []
> testkey
plainname: Key name? [] > testkey
x509country: Country code? [] > [...]
x509province: State or province? [] > [...]
```

```
x509locality: City or locality? [] > [...]
x509org: Organisation? [] > [...]
x509orgunit: Organisation unit? [] > [...]
x509dnscommon: Domain name? [] > [...]
x509email: Email address? [] > [...]
nvram: Blob in NVRAM (needs ACS)? (yes/no) [no] >
digest: Digest to sign cert req with? (md5, sha1, sha256, sha384, sha512)
[default sha256] >
key generation parameters:
operation       Operation to perform                generate
application     Application                         embed
protect         Protected by                        module
verify          Verify security of key              yes
type            Key type                            RSA
size            Key size                            2048
pubexp          Public exponent for RSA key (hex)
embedsavefile   Filename to write key to            testkey
plainname       Key name                            testkey
x509country     Country code                        [...]
x509province    State or province                   [...]
x509locality    City or locality                    [...]
x509org         Organisation                        [...]
x509orgunit     Organisation unit                   [...]
x509dnscommon   Domain name                         [...]
x509email       Email address                       [...]
nvram           Blob in NVRAM (needs ACS)           no
digest          Digest to sign cert req with        sha256
Key successfully generated.
Path to key: /opt/nfast/kmdata/local/key_embed_6d5706...
Path to CSR: <CURRENTFOLDER>/embed_6d5706..._req
Path to self-cert: <CURRENTFOLDER>/embed_6d5706..._selfcert
```

In the same folder as the self-cert there will also be a file called `testkey`.

Copy the files into the Apache installation using the following commands (adjust to the values you get):

```
cp <CURRENTFOLDER>/testkey /etc/pki/tls/private/testkey
cp <CURRENTFOLDER>/embed_6d5706..._selfcert /etc/pki/tls/certs/testkey_selfcert
```

In `/etc/httpd/conf.d/ssl.conf`, set

```
SSLCertificateFile /etc/pki/tls/certs/testkey_selfcert
SSLCertificateKeyFile /etc/pki/tls/private/testkey
SSLCryptoDevice chil
```

2.5.3. Open the firewall

```
firewall-cmd --zone=public --permanent --add-service=https
firewall-cmd --reload
```

2.5.4. Switch off SE Linux

If SE Linux is active, this might prevent Apache from loading our library. To switch it off:

```
setenforce 0
```

2.5.5. Start the HTTP daemon

```
service httpd start
```

`https://<yourapacheserver>` should work, and the certificate in the browser should show the information that was provided when creating the embed key above. For example:

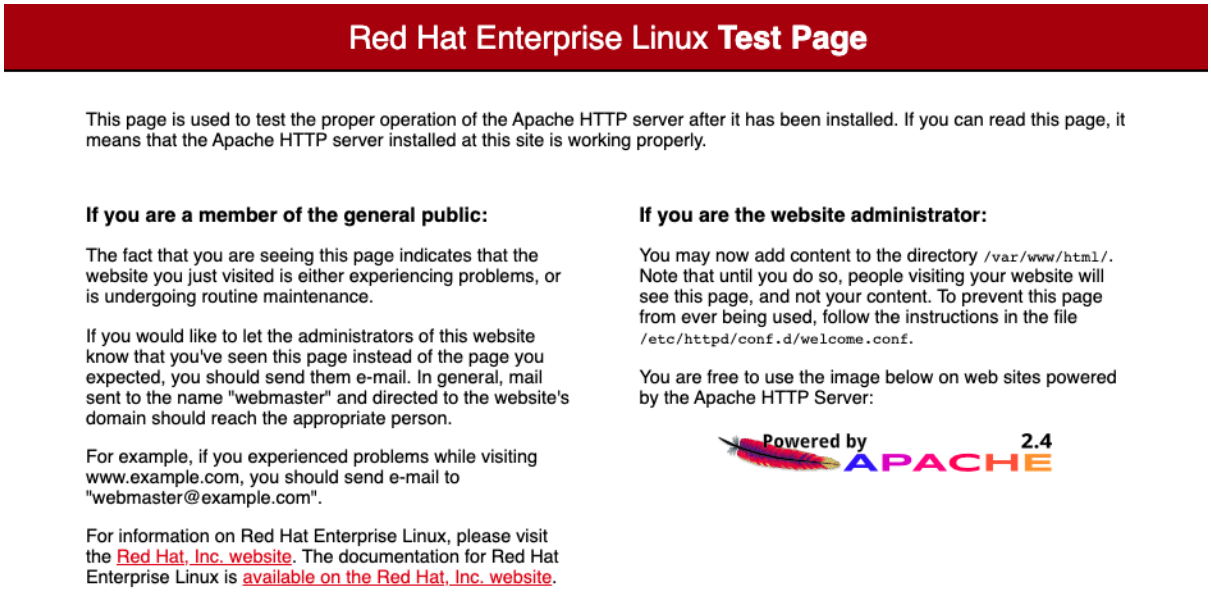

Figure 2.1 HTTPD successfully started

2.5.6. Test the connection

Test the connection with a command similar to:

```
openssl s_client -crlf -connect localhost:443 -CAfile testkey_selfcert.pem
openssl s_client -crlf -connect localhost:443 -CAfile <CURRENTFOLDER>/embed_6d5706.._
selfcert
```

Check the following messages and fields in the output:

- CONNECTED(00000003)

- depth
- Certificate chain information
- Server certificate information
- Session-ID
- Master-Key
- TLS session ticket:
- Verify return code: 0 (ok)

Example output:

```
# openssl s_client -crlf -connect localhost:443 -CAfile embed_6d5706..._selfcert
CONNECTED(00000003)
depth=[...]
verify return:1
---
Certificate chain
0 s:/C=[...]
i:/C=[...]
---
Server certificate
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
subject=[...]
issuer=[...]
---
No client certificate CA names sent
Peer signing digest: SHA512
Server Temp Key: ECDH, P-256, 256 bits
---
SSL handshake has read 1570 bytes and written 415 bytes
---
New, TLSv1/SSLv3, Cipher is ECDHE-...
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
Protocol : TLSv1.2
Cipher : ECDHE-...
Session-ID: [...]
Session-ID-ctx:
Master-Key: [...]
Key-Arg : None
Krb5 Principal: None
PSK identity: None
PSK identity hint: None
TLS session ticket lifetime hint: 300 (seconds)
TLS session ticket:
...
Start Time: 1579086822
Timeout : 300 (sec)
Verify return code: 0 (ok)
---
```

2.6. OCS protection

If OCS protection is required, create an OCS:

```
createocs -Q1/1 -Napacheocs -m 1
```

Leave the OCS in the card reader and generate an embed key as in Set up Apache to use the CHIL library, but choose the protection to be `token`.

The steps to copy certificates about is the same as for module-protected keys.

When you are starting Apache, you will have to preload the OCS so that the key can be used without the web server having to load it:

```
preload -f /var/run/httpd/preload -c apacheocs /usr/sbin/httpd -e debug -X
```

2.7. Softcard protection

If softcard protection is required, create a softcard:

```
ppmk -n apachesoft
```

Generate an embed key as in Set up Apache to use the CHIL library, but choose the protection to be `softcard`.

The steps to copy certificates about is the same as for module protected keys.

When you are starting Apache, you will have to preload the softcard so that the key can be used without the web server having to load it:

```
preload -f /var/run/httpd/preload -s apachesoft /usr/sbin/httpd -e debug -X
```

Chapter 3. Troubleshooting

If the logs produced by Apache do not lead to useful information, starting Apache with the following might lead to more information.

```
strace -f /usr/sbin/httpd 2> apache.trace
```

or

```
/usr/sbin/httpd -e debug -X
```

Chapter 4. Additional resources and related products

4.1. nShield Connect

4.2. nShield as a Service

4.3. Entrust products

4.4. nShield product documentation