Using Self-Signed Certificates with Nexus Repository Manager and Docker Daemon

Peter Lynch
Created: February 29, 2016 09:57 - Updated: January 25, 2018 10:20

Follow

Background

A private Docker registry such as Nexus Repository Manager 3 will require authentication from your users in order to publish docker images.

In order to use authentication, the Docker Daemon implementation enforces that the registry connection uses HTTPS.

Nexus 3 is not configured with HTTPS connectors by default as configuring it requires an SSL certificate to be generated and configured manually.

For testing purposes, it is common that an administrator may want to use a self-signed TLS certificate for Nexus HTTPS connectors.

Although self-signed certificates can encrypt HTTP communication and be generated quickly, they are generally considered untrustworthy because the certificate identity has not been signed/verified by a third party certificate authority (CA).

To work around the Docker Daemon considering the Nexus HTTPS as untrustworthy, the daemon has an option called ***--insecure-registry*** . By marking the Nexus hostname and port as insecure, the docker daemon does not validate the trustworthiness of the secure connection. The intent is to avoid errors that would normally occur when using an untrusted certificate.

However, the ***--insecure-registry*** flag has many unwanted side effects and can lead to obscure errors that would not be encountered otherwise.

Sonatype cannot recommend using the Docker ***--insecure-registry*** flag due to poorly defined semantics.

How To Generate a Self-Signed Certificate That Can Be Trusted By Docker Daemon

Rather than tell the docker daemon to not validate a self-signed certificate by using *--insecure-registry*, the better practice is to tell it to trust the self-signed certificate explicitly.

Docker provides documentation which describes using openssl to generate a CA and server self-signed certificates. However those instructions can lead to error messages where Docker will still not trust your self-signed certificate. Also Openssl is not implemented the same on all systems.

The steps listed below are an alternative, reliable way to generate a self-signed server certificate for Nexus in a platform neutral manner using Java keytool. The steps can be used to configure Docker to explicitly trust the connection to Nexus Repository Manager.

Step 1: Generate a self-signed server certificate for Nexus using keytool

First determine the host name and externally accessible IP address of Nexus that Docker will use. The host name and IP address will be embedded in the self-signed certificate so that host name certificate verification will not fail.

On the Nexus host, go to <AppDir>/etc/ssl directory. Generate a public private key pair for Nexus to use for any HTTPS connector.

In the command below:

- Replace ${NEXUS_DOMAIN} with the DNS name of your Nexus host. The host name must be resolvable from the host running the Docker daemon.
- Replace ${NEXUS_IP_ADDRESS} with the IP address of your Nexus host. The IP address must be routable to Nexus from the host running Docker daemon.

```
keytool -genkeypair -keystore keystore.jks -storepass password -keypass password -alias jetty -keyalg RSA -keysize 2048 -validity 5000 -dname "CN=*.${NEXUS_DOMAIN}, OU=Example, O=Sonatype, L=Unspecified, ST=Unspecified, C=US" -ext "SAN=DNS:${NEXUS_DOMAIN},IP:${NEXUS_IP_ADDRESS}" -ext "BC=ca:true"
```

You should end up with a file named keystore.jks in the local directory. The CN value specifies what is known as a wildcard certificate which will match any sub-domain.

Step 2: Configure Nexus to use the self-signed server certificate

For 3.0.x, follow the prerequisites of enabling SSL in Nexus.

For 3.1 or newer, see Inbound SSL - Configuring to Serve Content via HTTPS

In particular, make sure the SSLContextFactory is configured to point to the generated keystore file from step 1. Make sure these setting exist in jetty-https.xml:

Repository Manager 3.0.x Example: (<AppDir>/etc/jetty-https.xml)

```
<Set name="KeyStorePath"><Property name="karaf.etc"/>/ssl/keystore.jks</Set>
<Set name="KeyStorePassword">changeit</Set>
<Set name="KeyManagerPassword">changeit</Set>
<Set name="TrustStorePassword">changeit</Set>
```

Repository Manager 3.1.0 and Newer Example: (<AppDir>/etc/jetty/jetty-https.xml)

```
<Set name="KeyStorePath"><Property name="ssl.etc"/>/keystore.jks</Set>
<Set name="KeyStorePassword">changeit</Set>
<Set name="KeyManagerPassword">changeit</Set>
<Set name="TrustStorePassword">changeit</Set>
```

Restart Nexus to pick up any configuration changes.

Step 3: Configure Docker Daemon to trust the certificate

Use Oracle java `keytool` to retrieve and print the Nexus server certificate for the Nexus instance running at ${NEXUS_DOMAIN}:${SSL_PORT} :

```
keytool -printcert -sslserver ${NEXUS_DOMAIN}:${SSL_PORT} -rfc
```

This command will print an encoded certificate surrounded by `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----` lines.

If you have root access to the host running Docker daemon, the best option is to trust the Nexus server certificate at a system level. Copy the printed certificate and paste it into the file locations as described below.

Docker has general advice about how a unix based system can trust a certificate and we recommend to follow their advice.

A second option is to configure only Docker to trust the certificate on per host:port basis. Docker has specific advice on where certificates can be copied in order for them to be trusted automatically per host. The disadvantage is that for each Docker registry host:port accessed, a new certificate file must be added.

Was this article helpful? 0 out of 1 found this helpful

Have more questions? Submit a request

7 Comments

Date Votes

0 Avatar **Olivier Dupré** May 27, 2016 03:37

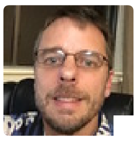
Hi Peter,

This is a very interesting article but... what about using Nexus behind NGinX?

Certificate generated with keytool cannot be used with NGinX. Is there anyway to use a self-signed certificate on NGinX and have docker being able to login to nexus? I followed your instructions, the ones on Docker site, ... and I just can't have docker login in Nexus. Should I bypass my NginX? (that would be weird) Should I setup a connector? ...

Cheers,

Olivier.

0  **Rich Seddon** May 27, 2016 08:45

The article is intended to cover SSL certificate installation in the default jetty connector only, it's beyond the scope of this article to cover SSL certificate installation in other products. But it will certainly work having nginx as the SSL termination for Nexus, we have lots of users who are doing this. You should be able to find instructions for using openssl to make a signing certificate, and a signed certificate easily. And be sure to use our nginx recommendations for reverse proxy with SSL termination:

https://books.sonatype.com/nexus-book/reference3/install.html#_example_reverse_proxy_ssl_termination_at_base_path

Regards,

Rich

0

Peter Lynch May 31, 2016 07:53

Olivier,

Certificate generated with keytool cannot be used with NGinX. Is there anyway to use a self-signed certificate on NGinX and have docker being able to login to nexus?

Look at step 4 and 5 of our article.

Later in that same article, it shows what nginx parameters need those certificate values.

Edited by Peter Lynch 2 years ago

0 Avatar

tenjhon July 21, 2016 04:13

hi Peter,

i configured docker repositories Disable redeploy ,and then ,when i push images, the fisrt image named jetty:latest pushed successfull,but when i push another image where named tomcat:latest,i get the error:"blob upload invalid: blob upload invalid". tow diffrent images are not allowed push ? what can i do ?

1

Peter Lynch July 21, 2016 12:51

Hi tenjhon - it is not clear what is wrong with the tomcat image here. If you think you've found a problem, please file an issue at https://issues.sonatype.org/browse/NEXUS with a support bundle and ROOT logger at DEBUG when the problem is reproduced.

0 Avatar

tenjhon July 21, 2016 21:05

It's beside the image name, I configured docker repositories Disable redeploy.

if the docker hosted repository is empoty,I push every image is successfull first,but when I push another image will fail.

if the docker hosted repository is not empoty,I push every image will fail.

when I configured docker repositories Allow redeploy,I push every images is OK.

I have reported the problem at https://issues.sonatype.org/browse/NEXUS

my English is so bad,hope you anderstand.

0

Peter Lynch

July 29, 2016 15:10

We are closing this article for comments. If you have a support license, please contact us by filing a support ticket.

If you do not have a support license, please use our nexus-users mailing list or other free resources.

Article is closed for comments.