BLACK DUCK HUB USING HTTPS: HOW TO MANAGE CERTIFICATES

Explore other articles and discussions on this topic.

Black Duck Hub using HTTPS: How to Manage Certificates

By default, the Black Duck Hub installed on Docker runs on HTTPS. The certificate used to run HTTPS is a self-signed certificate. What this means is that the certificate was created locally and was not signed by a recognized Certificate Authority. If you use this default certificate, you will notice that you need to make a security exception to log in to the Hub user interface. This is because your browser does not recognize the issuer of the certificate so it is not accepted by default.

Jun 14, 2018 · How To

HOW TO

By default, the Black Duck Hub installed on Docker runs on HTTPS. The certificate used to run HTTPS is a self-signed certificate. What this means is that the certificate was created locally and was not signed by a recognized Certificate Authority. If you use this default certificate, you will notice that you need to make a security exception to log in to the Hub user interface. This is because your browser does not recognize the issuer of the certificate so it is not accepted by default.

Generating a certificate:

You may want to generate a certificate for the Hub that is signed by a recognized Certificate Authority. To do so you would follow the below steps:

```
openssl genrsa -out <keyfile> <keystrength>
openssl req -new -key <keyfile> -out <CSRfile>
```

For Example:

```
openssl genrsa -out server.company.com.key 1024
openssl req -new -key server.company.com.key -out server.company.com.csr
```

You would then upload this .csr file to your Certificate Authority and they will provide you with a valid certificate that they have signed.

Note: The key will be required later so keep note of the location of the key that was generated above

Steps for Hub 4.0.0 and newer, Uploading certificate into Hub instance:

Docker-compose

You must first change the name of your certificate and key to the following:

Hub.crt -> WEBSERVER_CUSTOM_CERT_FILE

Hub.key -> WEBSERVER_CUSTOM_KEY_FILE

Next you will edit the docker-compose.yml file and edit the following line to point to your certificate-key pair

```
webserver:
  image: blackducksoftware/hub-nginx:4.0.0-SNAPSHOT
  ports: ['443:443']
  env_file: hub-webserver.env
  links: [webapp, cfssl]
  volumes: ['webserver-volume:/opt/blackduck/hub/webserver/security', '/directory/where/the/cert-key/is:/run/secrets']
```

If you know ahead of time that you will be using a custom certificate, run through this process before starting the Hub. Otherwise you will have to run a **docker-compose –p hub down** and then **docker-compose-p hub up -d**

Docker-swarm

After your stack is created you will need to run the following commands:

```
docker secret create <stack name>_WEBSERVER_CUSTOM_CERT_FILE <certificate file>
docker secret create <stack name>_WEBSERVER_CUSTOM_KEY_FILE <key file>
```

Then you will need to edit the docker-compose.yml file and add in the following property under the hub-webserver:

secrets:
 - WEBSERVER_CUSTOM_CERT_FILE
 - WEBSERVER_CUSTOM_KEY_FILE

Note: The below only applies if you are using your own or the provided self-signed certificate. If you generated a certificate signed by a CA then it will already be in the default keystore.

Steps for Prior Versions of the Hub, Uploading certificate into Hub instance:

Within the docker configuration package that is downloaded from Github, we provide a script to import the certificate signed by a CA.

From the base directory: `/hub-docker-3.7.1/docker-compose/bin/hub_webserver_use_custom_cert_key.sh`

Usage: `./hub_webserver_use_custom_cert_key.sh <crt-file-path> <key-file-path>`

You must provide a certificate in either .crt or .pem extension and then the .key previously generated for the csr request is also passed through. This script will import your certificate into the hub-webserver container and once the container comes back up your Hub server will now be running with the certificate that was just provided.

Caution: Using a Self-Signed Certificate or Locally Signed Certificate

Note: The next two sections only apply if you are using your own or the provided self-signed certificate. If you generated a certificate signed by a CA then it will already be in the default keystore.

Using the scanner with HTTPS:

Since Hub 3.6 the Hub Scanner will now import the certificate into the Hub Scanner CLI keystore for you. When you run the scanner to your Hub server running HTTPS you will get the following prompts:

```
./scan.cli.sh (http://scan.cli.sh) --username sysadmin --host sup-hub02.dc1.lan --port 443 --scheme https
/Users/sgraham/Downloads/openssl-SSLeay_0_8_1b.zip --project SSL --release 1.0

Cannot verify the identity of the server "hub-webserver". The certificate for this server is invalid. Would you like
to connect anyway?
[1] Cancel          [2] Continue        [3] Always Connect
Enter selection [1]:
```

The scanner could not verify the certificate because it is a self-signed certificate and not issued by a CA. If you uploaded you certificate signed by a Certificate Authority you would not see this message as your certificate would already be in the default cacerts keystore within the scanner.

Choosing 1 will cancel the scan, choosing 2 will use the cert for this one scan, and choosing 3 will import the certificate permanently into your scanner keystore.

Using HTTPS with integrations:

Since the Hub by default has a self-signed certificate your CI server server will not recognize the certificate. You will have to run the following commands to get the public certificate key and then import it into your Java Home keystore for your server:

```
openssl s_client -connect hubserver:443 2>/dev/null | openssl x509 -in /dev/stdin > hub.crt

 keytool -import -trustcacerts -keystore $JAVA_HOME/jre/lib/security/jssecacerts -file /path/to/certificate -alias
hub
```

If you are using Jenkins, Bamboo or TeamCity, the plugin does not currently prompt you to import the certification. This will cause your first build to fail since the plugin cannot pick one of the above options during the build. Once the build fails its first time, you will have the Scanner downloaded into your tool directory. You will have to manually import your certificate into the scanner keystore. In this case, you will only need the public certificate key. The below commands will get the public certificate and import it into the keystore:

```
openssl s_client -connect hubserver:443 2>/dev/null | openssl x509 -in /dev/stdin > hub.crt

 keytool -import -trustcacerts -keystore $SCANNER_HOME/jre/lib/security/jssecacerts -file /path/to/certificate -alias
hub
```

Now that the certificate is added in, your next build will go through fine (in accordance with HTTPS).

ATTACHMENT

PRODUCT ARTICLE TYPE

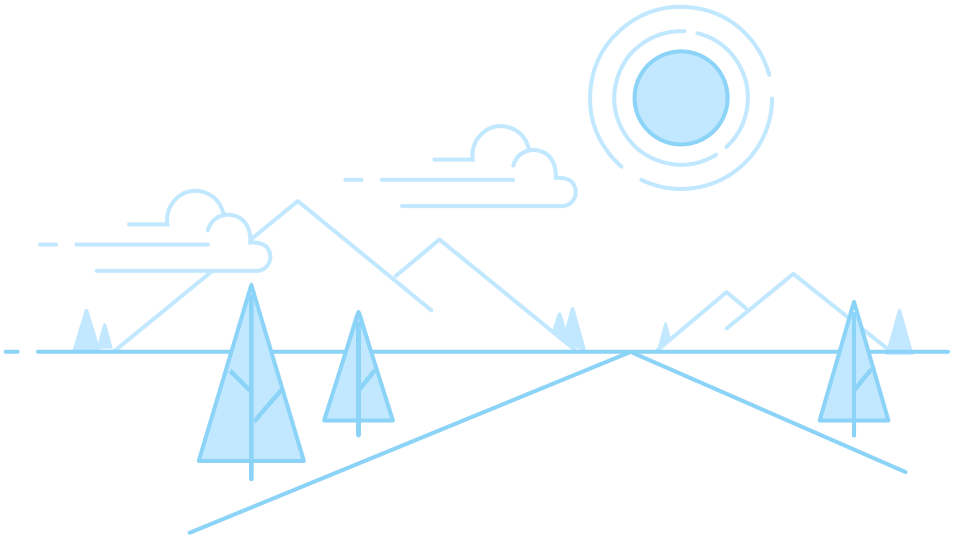
Black Duck Hub

Black Duck - Hub (/s/topic/0TO34000000UGyT...

Sort by:

Latest Posts

Search this feed...



Nothing here yet?

Login to comment on this post

Did this content resolve your issue?

Yes No

RELATED ARTICLES

Integrations Documentation - Hub Detect (/s/article/Integrations-Documentation-Hub-Detect) 1.25K

Integrations Documentation - Black Duck JIRA Plugin (/s/article/Integrations-Documentation-Black-Duck-JIRA-Plugin) 123

Integrations Documentation - Black Duck Alert (/s/article/Integrations-Documentation-Black-Duck-Alert) 185

Integrations Documentation - Black Duck Artifactory Plugin (/s/article/Integrations-Documentation-Black-Duck-Artifactory-Plugin) 187

Integrations Documentation - Black Duck Docker Inspector (/s/article/Integrations-Documentation-Black-Duck-Docker-Inspector) 280

TRENDING ARTICLES

Coverity Tutorial: Introduction to Coverity [Video]
(/s/article/Coverity-Tutorial-Introduction-to-Coverity)

Coverity Tutorial: Basic Workflow [Video]
(/s/article/Coverity-Tutorial-Basic-Workflow)