Verify repository client with certificates

Estimated reading time: 2 minutes

In Running Docker with HTTPS (https://docs.docker.com/engine/security/https/), you learned that, by default, Docker runs via a non-networked Unix socket and TLS must be enabled in order to have the Docker client and the daemon communicate securely over HTTPS. TLS ensures authenticity of the registry endpoint and that traffic to/from registry is encrypted.

This article demonstrates how to ensure the traffic between the Docker registry server and the Docker daemon (a client of the registry server) is encrypted and properly authenticated using *certificate-based client-server authentication*.

We show you how to install a Certificate Authority (CA) root certificate for the registry and how to set the client TLS certificate for verification.

Understand the configuration

A custom certificate is configured by creating a directory under `/etc/docker/certs.d` using the same name as the registry's hostname, such as `localhost`. All `*.crt` files are added to this directory as CA roots.

> **Note**: As of Docker 1.13, on Linux any root certificates authorities are merged with the system defaults, including as the host's root CA set. On prior versions of Docker, and on Docker Enterprise Edition for Windows Server, the system default certificates are only used when no custom root certificates are configured.

The presence of one or more `<filename>.key/cert` pairs indicates to Docker that there are custom certificates required for access to the desired repository.

> **Note**: If multiple certificates exist, each is tried in alphabetical order. If there is a 4xx-level or 5xx-level authentication error, Docker continues to try with the next certificate.

The following illustrates a configuration with custom certificates:

```
/etc/docker/certs.d/        <-- Certificate directory
└── localhost:5000          <-- Hostname:port
   ├── client.cert          <-- Client certificate
   ├── client.key           <-- Client key
   └── ca.crt               <-- Certificate authority that signed
                                the registry certificate
```

The preceding example is operating-system specific and is for illustrative purposes only. You should consult your operating system documentation for creating an os-provided bundled certificate chain.

Create the client certificates

Use OpenSSL's `genrsa` and `req` commands to first generate an RSA key and then use the key to create the certificate.

```
$ openssl genrsa -out client.key 4096
$ openssl req -new -x509 -text -key client.key -out client.cert
```

> **Note**: These TLS commands only generate a working set of certificates on Linux. The version of OpenSSL in macOS is incompatible with the type of certificate Docker requires.

Troubleshooting tips

The Docker daemon interprets `.crt` files as CA certificates and `.cert` files as client certificates. If a CA certificate is accidentally given the extension `.cert` instead of the correct `.crt` extension, the Docker daemon logs the following error message:

```
Missing key KEY_NAME for client certificate CERT_NAME. CA certificates should u
```

If the Docker registry is accessed without a port number, do not add the port to the directory name. The following shows the configuration for a registry on default port 443 which is accessed with `docker login my-https.registry.example.com`:

```
/etc/docker/certs.d/
└── my-https.registry.example.com          <-- Hostname without port
   ├── client.cert
   ├── client.key
   └── ca.crt
```

Related information

- Use trusted images (https://docs.docker.com/engine/security/)
- Protect the Docker daemon socket (https://docs.docker.com/engine/security/https/)

Usage (https://docs.docker.com/glossary/?term=Usage), registry (https://docs.docker.com/glossary/?term=registry), repository (https://docs.docker.com/glossary/?term=repository), client (https://docs.docker.com/glossary/?term=client), root (https://docs.docker.com/glossary/?term=root), certificate (https://docs.docker.com/glossary/?term=certificate), docker (https://docs.docker.com/glossary/?term=docker), apache (https://docs.docker.com/glossary/?term=apache), ssl (https://docs.docker.com/glossary/?term=ssl), tls (https://docs.docker.com/glossary/?term=tls), documentation (https://docs.docker.com/glossary/?term=documentation), examples (https://docs.docker.com/glossary/?term=examples), articles (https://docs.docker.com/glossary/?term=articles), tutorials (https://docs.docker.com/glossary/?term=tutorials)