SECURE CONFIGURATION

All XYZ software is secured using certificates that verify configuration files during installation to avoid loading malicious configuration at customer sites. The following steps are to be used by developers to sign any configuration files delivered to production.

ROOT CERTIFICATE AUTHORITY

A self-signed root certificate authority (CA) was generated at XYZ company with a key size of 4K. The certificate is valid for 20 years and renewable every 10 years. The password is physically locked in a vault.

PRODUCT INTERMEDIATE CA

Each software product should have a security product owner who generates and owns the intermediate CA. The intermediate CA is used to sign development certificates. To create an intermediate CA, the security product owner should perform the following steps:

1. Using the root CA, create an intermediate key with a size of 4K.
2. Generate an intermediate certificate, valid for 5 years, renewable every 3 years.
3. Create a certificate chain.
4. Sign development Certificate Signing Requests (CSRs) using the intermediate CA.

DEVELOPER CERTIFICATES

Each developer on the software product must have their own certificate. To create a certificate, perform the following steps.

1. Generate your keys using OpenSSL.
2. Create a Certificate Signing Request.
3. Send the CSR in signed email to your security product owner.
4. Receive a signed certificate and CA chain valid for one year.
5. Import the signed certificate and CA chain into your keystore.
6. Use the jarsigner tool or the Maven jarsigner plugin to sign your configuration jar files.
    - It is recommended that signing is set up to be done as part of automated builds.
    - Use a local timestamp authority to reduce load during builds.
7. Include installation tests to verify valid and invalid configuration files. See next section.

VERIFICATION

Signed jar files are included as part of the software delivery. During deployment, the installation tools verify the authenticity of the configuration files as follows.

1. Recompute the digest of each jar and compare with the digest recorded in the manifest to ensure that the contents of the jar file have not changed since it was signed.
2. Verify the signer with the certificate chain.
3. Validate the signing timestamp against the certificate validity.
4. If a file fails verification, it will not be loaded.
5. The audit log includes information of the signer (common name and email address).