CEM MiNiFi Java Agent 2.24.02

Securing MiNiFi Java Agent

Date published: 2022-07-28
Date modified: 2024-04-16

CLOUDERA

https://docs.cloudera.com/

Legal Notice

Contents

MiNiFi Java Agent authentication

Learn about the properties that you need to set to configure TLS for the MiNiFi Java Agent.

To configure the TLS context for MiNiFi Java, including client certificates and trust settings, edit the following properties in conf/bootstrap.conf:

```
# Security Properties #
# These properties take precedence over any equivalent properties specified
in flow.json.raw file #
nifi.minifi.security.keystore=/path/to/keystore.jks
nifi.minifi.security.keystoreType=JKS
nifi.minifi.security.keystorePasswd=password
nifi.minifi.security.keyPasswd=password
nifi.minifi.security.truststore=/path/to/truststore.jks
nifi.minifi.security.truststoreType=JKS
nifi.minifi.security.truststorePasswd=password
nifi.minifi.security.ssl.protocol=TLSv1.2
# Properties for encrypting keystore and truststore passwords
nifi.minifi.sensitive.props.key=passwordOfAtLeast12Characters
nifi.minifi.sensitive.props.algorithm=NIFI_PBKDF2_AES_GCM_256

# Properties needed only if the agent is communicating with EFM
c2.security.truststore.location=
c2.security.truststore.password=
c2.security.truststore.type=
c2.security.keystore.location=
c2.security.keystore.password=
c2.security.keystore.type=
c2.security.need.client.auth=
```

Encrypting sensitive properties in configuration files

Learn how to encrypt sensitive properties in the conf/bootstrap.conf file using the encrypt-config command line tool, invoked in the minifi-toolkit as ./bin/encrypt-config.sh or bin\encrypt-config.bat.

About this task

This tool reads plain text sensitive configuration values from the bootstrap.conf file and encrypts each value using a random encryption key. It replaces the plain values with the protected value in the same file or writes to a new bootstrap.conf file, if specified. Additionally, it can be used to encrypt unencrypted sensitive properties (if any) in the flow.json.raw file.

To enable this functionality, ensure that the nifi.minifi.sensitive.props.key and nifi.minifi.sensitive.props.algorithm properties are provided in bootstrap.conf.

Example

The following example shows how the tool works with existing values in the bootstrap.conf file:

```
nifi.sensitive.props.key=thisIsABadSensitiveKeyPassword
nifi.sensitive.props.algorithm=NIFI_PBKDF2_AES_GCM_256
nifi.sensitive.props.additional.keys=

nifi.security.keystore=/path/to/keystore.jks
nifi.security.keystoreType=JKS
nifi.security.keystorePasswd=thisIsABadKeystorePassword
```

```
nifi.security.keyPasswd=thisIsABadKeyPassword
nifi.security.truststore=
nifi.security.truststoreType=
nifi.security.truststorePasswd=
c2.security.truststore.location=
c2.security.truststore.password=thisIsABadTruststorePassword
c2.security.truststore.type=JKS
c2.security.keystore.location=
c2.security.keystore.password=thisIsABadKeystorePassword
c2.security.keystore.type=JKS
```

Enter the following arguments when using the tool:

```
encrypt-config.sh -b %MINIFI_HOME_DIR%/conf/bootstrap.conf
```

As a result, the bootstrap.conf file is overwritten with protected properties and sibling encryption identifiers (aes/gcm/256, the currently supported algorithm):

```
nifi.sensitive.props.key=4OjkrFywZb7BlGz4||Tm9pg0jV4TltvVKeiMlm9zBsqmtmYUA2Q
kzcLKQpspyggtQuhNAkAla5s2695A==
nifi.sensitive.props.key.protected=aes/gcm/256
nifi.sensitive.props.algorithm=NIFI_PBKDF2_AES_GCM_256
nifi.sensitive.props.additional.keys=
nifi.security.keystore=/path/to/keystore.jks
nifi.security.keystoreType=JKS
nifi.security.keystorePasswd=iXDmDCadoNJ3VotZ||WvOGbrii4Gk0vr3b6mDstZg+NE0B
PZUPk6LVqQlf2Sx3G5XFbUbUYAUz
nifi.security.keystorePasswd.protected=aes/gcm/256
nifi.security.keyPasswd=199uUUgpPqB4Fuoo||KckbW7iu+HZf1r4KSMQAFn8NLJK+CnUuay
qPsTsdM0Wxou1BHg==
nifi.security.keyPasswd.protected=aes/gcm/256
nifi.security.truststore=
nifi.security.truststoreType=
nifi.security.truststorePasswd=
c2.security.truststore.location=
c2.security.truststore.password=0pHpp+l/WHsDM/sm||fXBvDAQ1BXvNQ8b4EHKa1Gsp
sLx+UD+2EDhph0HbsdmgpVhEv4qj0q5TDo0=
c2.security.truststore.password.protected=aes/gcm/256
c2.security.truststore.type=JKS
c2.security.keystore.location=
c2.security.keystore.password=j+80L7++RNDf9INQ||RX/QkdVFwRos6Y4XJ8YSUWoI3W
5Wx50dyw7HrAA84719SvfxA9eUSDEA
c2.security.keystore.password.protected=aes/gcm/256
c2.security.keystore.type=JKS
```

Additionally, the bootstrap.conf file is updated with the encryption key as follows:

```
minifi.bootstrap.sensitive.key=c92623e798be949379d0d18f432a57f1b74732141be32
1cb4af9ed94aa0ae8ac
```

Sensitive configuration values are encrypted by the tool by default, but you can encrypt additional properties, if desired. To encrypt additional properties, specify them as comma-separated values in the minifi.sensitive.props.additional.keys property.

Note:

If the bootstrap.conf file already contains valid protected values, those property values are not modified by the tool.

Example

The following example shows how to encrypt non-encrypted sensitive properties in the flow.json.raw file using the tool.

```
nifi.sensitive.props.key=sensitivePropsKey
nifi.sensitive.props.algorithm=NIFI_PBKDF2_AES_GCM_256
```

Enter the following arguments when using the tool:

```
encrypt-config.sh -x -f %MINIFI_HOME_DIR%/conf/flow.json.raw
```

As a result, the flow.json.raw file is overwritten with encrypted sensitive properties.

The algorithm uses property descriptors in the flow.json.raw file to determine if a property is sensitive or not. If that information is missing, no properties will be encrypted, even if defined as sensitive in the agent manifest.