Breaking Change: 4.10 KeyStore Permission Checks

Overview

There is an existing API in baja in the javax.baja.security.crypto package for managing certificates and keys in the Niagara keystores. This allows third party developers unrestricted access to read or manipulate the keystores. This breaking change proposal is to add new permission checks to better protect the keystores and require third party developers to request a permission to gain access to them. This breaking change is for Niagara 4.11.

Impact of Changes

Third party modules that use the certificate management API will start to see an AccessControlException similar to the one below when attempting to read or write the keystores.

java.security.AccessControlException: access denied ("com.tridium.nre.security.KeyStorePermission" "keyStore" "write")

The following method calls will now be protected by a permission check:
- ITrustStore.aliases(),
- ITrustStore.containsAlias(String),
- ITrustStore.getCertificate(String)
- ITrustStore.getCertificateAlias(X509Certificate)
- ITrustStore.getCertificateChain(String)
- ITrustStore.getCreationDate(String)
- ITrustStore.isCertificateEntry(String)
- ITrustStore.isKeyEntry(String)
- ITrustStore.size()
- ITrustStore.getCertificates()
- ITrustStore.findCertificate(X509Certificate)
- IKeyStore.getKey(String, char[])
- ITrustStore.setCertificateEntry(String, X509Certificate)
- ITrustStore.save()
- IKeyStore.setKeyEntry(String, byte[], X509Certificate[])
- IKeyStore.setKeyEntry(String, Key, char[], X509Certificate[])
- ITrustStore.deleteEntries(String[])

Note that read access on the user trust store and system trust store will be granted to all modules by default. If your module is only reading from these trust stores, no additional action is required.

Remediation

To resolve the issue, the module must add the KEY_STORE permission group with the relevant parameters to the module-permissions.xml file. Required parameters are:

• keystores: The keystore you are requesting permissions for. Can be one or more valid keystore name, separated by commas. Valid options are userTrustStore, userUntrustedStore, systemTrustStore, and userKeyStore. A wildcard of '*' can be used for all keystores.
• actions: read, write, or all.

Below is an example of KEY_STORE permissions being requested in a module-permissions.xml file that allows read on all keystores, and write on the user trust store:

```
...
<req-permission>
   <name>KEY_STORE</name>
   <purposeKey>keyStore.purpose</purposeKey>
   <parameters>
     <parameter name="keystores" value="*"/>
     <parameter name="actions" value="read"/>
 </parameters>
</req-permission>
<req-permission>
   <name>KEY_STORE</name>
   <purposeKey>keyStore.purpose</purposeKey>
   <parameters>
     <parameter name="keystores" value="userTrustStore"/>
     <parameter name="actions" value="write"/>
 </parameters>
</req-permission>
...
```

In addition to requesting the necessary KEY_STORE permissions, it may also be necessary to add a doPrivileged() block depending on the modules on the calling stack:

```
Key key = AccessController.doPrivileged((PrivilegedExceptionAction<Key>) () -> {
  return CertManagerFactory.getInstance().getKeyStore().getKey("myKey", new char[0]);
});
```

Program Objects

Since program objects do not support requesting additional permissions, program objects will no longer be able to perform actions on the KeyStore that require more than the default permissions (read on user trust store and system trust store). If you have a program object that is doing more than this, you will need to either convert it to a program module or a fully-fledged module and request the necessary permissions.