Java Cryptography Extension KeyStore (JCEKS) Setup

CBOM Secure · Keystore Sensor (Discover_Keystore)

Overview

The CBOM Secure Discover_Keystore sensor scans JCEKS files to inventory cryptographic assets. JCEKS extends JKS with SecretKeyEntry support, storing symmetric keys (AES, DES3, HMAC) alongside private keys and certificates. The sensor is read-only and reports metadata only (alias, algorithm, size, entry type, creation date) — never raw secret-key bytes.

Prerequisites

- A JRE/JDK 8+ on the host, with keytool on PATH.
- One or more .jceks files on the host or a network-accessible path.
- The storepass (and keypass if different) for each file.
- The CBOM agent installed with read access to the files.

Step-by-Step Guide

Step 1: Locate JCEKS Files

```
find /opt -name "*.jceks" -type f 2>/dev/null
find /opt /etc /var /home \( -name "*.jceks" -o -name "*.jks" \) -type f 2>/dev/null
setfacl -m u:cbom-agent:r /opt/app/config/secrets.jceks    # grant read if needed
```

Step 2: Inspect a Keystore

```
keytool -list \
  -keystore /opt/app/config/secrets.jceks \
  -storetype jceks -storepass changeit
```

SecretKeyEntry items show only alias and creation date — key bytes are never displayed by keytool or the sensor.

Step 3: Supply the Keystore Password Securely

```
cbom secret set --name jceks_storepass_app --value "changeit"
# or an environment variable referenced as env://CBOM_JCEKS_STOREPASS
```

Step 4: Configure the CBOM Secure Sensor

```
sensors:
  - name: Discover_Keystore
    enabled: true
    type: keystore
    schedule: "0 2 * * *"
```

```
  targets:
    - id: app-secrets-jceks
      path: /opt/app/config/secrets.jceks
      storetype: jceks
      storepass: secret://jceks_storepass_app
  discovery:
    extract_entry_types: [SecretKeyEntry, PrivateKeyEntry, TrustedCertificateEntry]
    report_key_material: false        # always false
    include_metadata: [alias, entry_type, algorithm, key_size, creation_date]
  output:
    format: cbom-json
    tags: { keystore_format: jceks }
cbom sensor apply -f /etc/cbom/sensors/keystore.yaml
```

Step 5: Validate

```
cbom sensor run --name Discover_Keystore --target app-secrets-jceks
journalctl -u cbom-agent -n 100 --no-pager | grep Discover_Keystore
```

Confirm SecretKeyEntry and PrivateKeyEntry discoveries with algorithm and size, then verify under Assets → Cryptographic Keys filtered by Format: JCEKS — with no raw key material in any field.

Common Errors

Keystore was tampered with, or password was incorrect

Cause: The storepass does not match, or the file is corrupted.

Resolution: Verify with keytool -list; update the CBOM secret if rotated; restore from backup if corrupted.

JCEKS not found (KeyStoreException)

Cause: The JRE lacks the SunJCE provider, or storetype is wrong.

Resolution: Confirm SunJCE is available and set storetype: jceks (lowercase).

Permission denied

Cause: The agent user cannot read the JCEKS file.

Resolution: Grant read via setfacl -m u:cbom-agent:r and re-run.

Security Recommendations

- Never store keystore passwords in plaintext — use secret:// or env:// references and rotate regularly.
- Restrict JCEKS file permissions to the owning app user and the agent; remove world-read.
- Audit contents with keytool -list -v to find stale/weak entries.
- Prefer AES-256 over DES/3DES/RC2; treat flagged deprecated algorithms as remediation items.
- Store JCEKS files on encrypted volumes; never in version control.

Conclusion

The Discover_Keystore sensor provides read-only discovery of symmetric keys, private keys, and trusted certificates in JCEKS files — ensuring symmetric key usage, often the least visible asset class, is fully represented in CBOM Secure without exposing raw material.