UBER Keystore Setup

CBOM Secure · Keystore Sensor (Discover_Keystore)

Overview

This guide configures the CBOM Secure Discover_Keystore sensor to inventory cryptographic material in UBER-format keystore files (.ubr). UBER is a BouncyCastle proprietary format using Twofish encryption and SHA-512 for key wrapping and integrity - stronger than BKS - found in legacy BouncyCastle deployments. The sensor reads .ubr files read-only via the BouncyCastle JCE provider and extracts:

- Private keys and their associated certificates
- Public key certificates (trusted entries)
- Secret key entries
- Alias names, key types, key sizes, and validity periods
- Signature algorithms and issuer/subject DNs

Prerequisites

- A running CBOM instance with the Discover_Keystore sensor licensed.
- Read-only filesystem access to the host holding the .ubr files.
- The BouncyCastle provider JAR (bcprov-jdk18on-*.jar) on the sensor host.
- JDK 8 or later (for keytool inspection).
- The store password for each keystore, plus any key-level passwords.
- Read permission on the directories containing .ubr files.

Step-by-Step Guide

Step 1: Locate UBER Keystore Files

```
find /opt /usr/local /home /var/lib /etc -name '*.ubr' -type f 2>/dev/null
ls -la /opt/myapp/security/store.ubr        # confirm sensor account can read
```

Step 2: Verify BouncyCastle Provider Availability

```
find /opt /usr/local/lib /usr/share/java -name 'bcprov-*.jar' 2>/dev/null
# if absent, download from the official distribution and verify its checksum:
wget https://downloads.bouncycastle.org/java/bcprov-jdk18on-178.jar \
  -O /opt/cbom/lib/bcprov-jdk18on-178.jar
```

Confirm the provider can open a target keystore before configuring the sensor:

```
keytool -list -keystore /opt/myapp/security/store.ubr \
  -storetype UBER -storepass changeit \
  -provider org.bouncycastle.jce.provider.BouncyCastleProvider \
  -providerpath /opt/cbom/lib/bcprov-jdk18on-178.jar
```

Step 3: Prepare Keystore Credentials

Register each store password (and any per-alias key password) as a named secret in the CBOM secrets vault (UI > Settings > Secrets) rather than hardcoding it - e.g. uber-keystore-myapp-pass.

Step 4: Configure the CBOM Secure Sensor

```
sensor:
  name: Discover_Keystore
  enabled: true
  scan_id: uber-keystore-scan-prod-001
  keystore:
    format: UBER
    provider: org.bouncycastle.jce.provider.BouncyCastleProvider
    provider_jar_path: /opt/cbom/lib/bcprov-jdk18on-178.jar
  discovery:
    paths:
      - /opt/myapp/security/store.ubr
      - /var/lib/bc-app/keys/root.ubr
    path_patterns: [ '/opt/**/*.ubr', '/usr/local/**/*.ubr' ]
    recursive: true
    follow_symlinks: false
  credentials:
    default_store_password:
      secret_ref: uber-keystore-default-pass
    per_file_credentials:
      - path: /opt/myapp/security/store.ubr
        store_password: { secret_ref: uber-keystore-myapp-pass }
  output:
    include_certificate_chain: true
    redact_private_key_material: true
  schedule:
    cron: "0 2 * * *"
```

Save to /etc/cbom/sensors/uber-keystore-prod.yaml and restart: systemctl restart cbom-sensor.

Step 5: Validate

```
tail -f /var/log/cbom/uber-keystore-sensor.log       # look for 'Opened UBER keystore ... Scan complete'
cbom-ctl sensor trigger --scan-id uber-keystore-scan-prod-001
```

Confirm assets under Assets > Cryptographic Keys filtered by source uber-keystore-scan-prod-001, and cross-check aliases and fingerprints against the keytool output.

Common Errors

KeyStoreException: UBER not found

Cause: The BouncyCastle JAR is not on the classpath or provider_jar_path is wrong/unreadable.

Resolution: Confirm the JAR exists and is readable by the sensor account, correct provider_jar_path, and restart the sensor.

Keystore was tampered with, or password was incorrect

Cause: The store password is wrong or the file is corrupted (UBER uses SHA-512 integrity).

Resolution: Test the password with keytool; if correct but still failing, restore from backup, then update the vault secret and re-trigger.

UnrecoverableKeyException: Failed to recover key

Cause: A specific alias uses a distinct key password not matching the store password.

Resolution: Identify the failing alias in the log, obtain its key password, and add a key_passwords entry for that path/alias, then restart and re-trigger.

Security Recommendations

- Never hardcode passwords in YAML - reference the CBOM secrets vault or an external manager.
- Restrict .ubr file permissions to 640 or tighter, owned by the application user.
- Validate the BouncyCastle JAR checksum against the published value before deployment.
- Rotate keystore passwords on a defined schedule and update the vault after rotation.
- Audit sensor access logs periodically to confirm scan activity matches expectations.

Conclusion

By locating .ubr files, confirming BouncyCastle provider availability, and supplying credentials through the CBOM secrets vault, the Discover_Keystore sensor continuously and securely inventories all keys and certificates in UBER keystores - supporting compliance, certificate lifecycle management, and cryptographic agility for legacy BouncyCastle deployments.