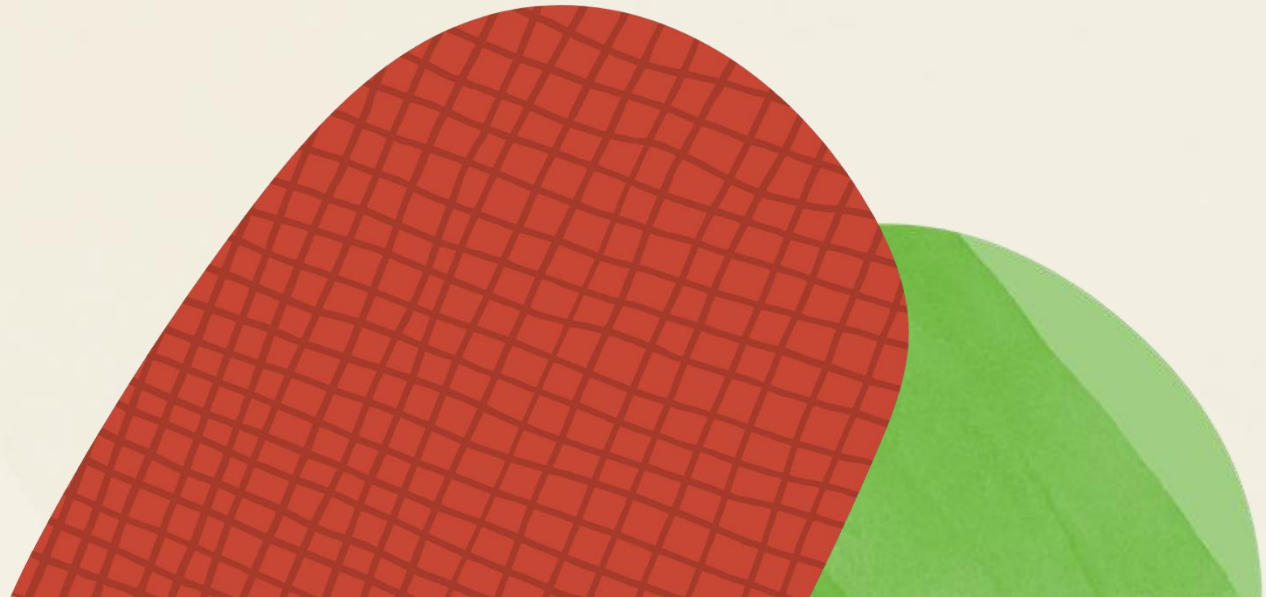
Monitoring Security Operations with JDK Flight Recorder Events

Seán Coffey, Oracle
FOSDEM 2025



JDK Flight Recorder (JFR) Overview

- Low cost monitoring and profiling framework embedded directly in the JVM
- Production use. Typical 1% overhead
- Events form the building blocks of the data output. Captured to .jfr file
- Traditionally used for JVM, low level monitoring:
  Memory, GC, threads, locks, etc.
- New events added to help monitor JDK usage at the security and core library API level

Launching JFR

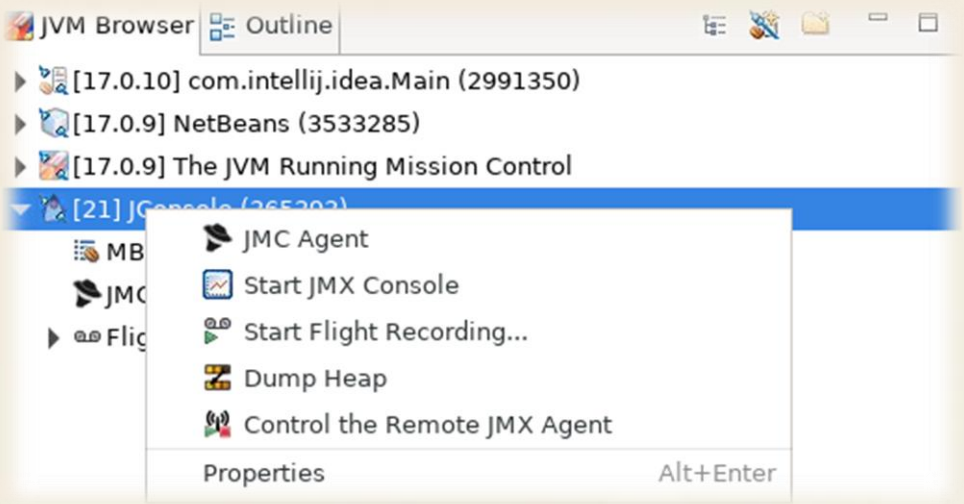
- Flexible options for capturing JFR data:
- Via command line. e.g. : -XX:StartFlightRecording=dumponexit=true MyApp

Launching JFR

- Flexible options for capturing JFR data:
- Via command line. e.g. : -XX:StartFlightRecording=dumponexit MyApp
- Via jcmd : e.g. jcmd <pid> JFR.start

Launching JFR

- Flexible options for capturing JFR data:
- Via command line. e.g. : -XX:StartFlightRecording=dumponexit MyApp
- Via jcmd : e.g. jcmd <pid> JFR.start
- Java Mission Control (JMC)



JFR Events targeting JDK core libraries

- More events introduced to give visibility of core library level usage
- Examples:

Event Name	Introduced	Description
jdk.Deserialization	JDK 17	Record deserialization of objects
jdk.ProcessStart	JDK 15	Record details of processes started via ProcessBuilder
jdk.VirtualThreadStart jdk.VirtualThreadEnd	JDK 19	Indicate when a virtual thread starts and ends
jdk.Shutdown	JDK 11	Capture details of code calling System.exit
jdk.InitialEnvironmentVariable	-	Details of all JVM environment variables
Security/Crypto events	JDK 12+	…

Motivation for new JFR Crypto Events

- Monitor security configuration and actions of Java runtime
  - Is your environment safe ?
- Allow monitoring of application code via remote attach mechanism
- Determine impact of JDK Cryptographic Roadmap changes on applications
- https://www.java.com/en/jre-jdk-cryptoroadmap.html
- Binary data to allow efficient queries on complex stacks

JFR Crypto Events

Event Name	Introduced	Enabled by Default ?
jdk.SecurityPropertyModification	JDK 12	No
jdk.TLSHandshake	JDK 12	No
jdk.X509Certificate	JDK 12	No
jdk.X509Validation	JDK 12	No
jdk.InitialSecurityProperty	JDK 20	Yes
jdk.SecurityProviderService	JDK 20	No

Enabling JFR Crypto Events

- Default configuration file at $JDK/lib/jfr/default.jfc
- Extra profiling: $JDK/lib/jfr/profile.jfc
- Specify custom configuration file via start up arguments. E.g. :
- -XX:StartFlightRecording=filename=/home/demo.jfr,**settings=/home/verbose.jfc**

Enabling JFR Crypto Events

- Default configuration file at $JDK/lib/jfr/default.jfc
- Extra profiling: $JDK/lib/jfr/profile.jfc
- Specify custom configuration file via start up arguments. E.g. :
- -XX:StartFlightRecording=filename=/home/demo.jfr,**settings=/home/verbose.jfc**

```
<event name="jdk.SecurityProviderService">
      <setting name="enabled">false</setting>
      <setting name="stackTrace">true</setting>
</event>
```


Enabling JFR Crypto Events

- Default configuration file at $JDK/lib/jfr/default.jfc
- Extra profiling: $JDK/lib/jfr/profile.jfc
- Specify custom configuration file via start up arguments. E.g. :
- -XX:StartFlightRecording=filename=/home/demo.jfr,**settings=/home/verbose.jfc**

```
<event name="jdk.SecurityProviderService">
      <setting name="enabled">false</setting>
      <setting name="stackTrace">true</setting>
</event>
```

jdk.SecurityPropertyModification

- Records details of each Security.setProperty(String, String) call

```
#jfr print --events jdk.SecurityPropertyModification  --stack-depth 100 prop.jfr
jdk.SecurityPropertyModification {
  startTime = 10:35:05.295 (2025-01-15)
  key = "jdk.tls.disabledAlgorithms"
  value = "TLSv1"
  eventThread = "main" (javaThreadId = 3)
  stackTrace = [
    HttpsClient.main(String[]) line: 95
  ]
}
```


jdk.TLSHandshake

- Records details of each TLS handshake from JSSE provider

```
jdk.TLSHandshake {
  startTime = 12:06:02.947 (2025-01-14)
  peerHost = "www.youtube.com"
  peerPort = 443
  protocolVersion = "TLSv1.3"
  cipherSuite = "TLS_AES_256_GCM_SHA384"
  certificateId = 1533204685
  eventThread = "main" (javaThreadId = 3)
  stackTrace = [
    sun.security.ssl.Finished$T13FinishedProducer.onProduceFinished(ClientHandshakeContext, SSLHandshake$HandshakeMessage) line: 767
    sun.security.ssl.Finished$T13FinishedProducer.produce(ConnectionContext, SSLHandshake$HandshakeMessage) line: 672
    sun.security.ssl.SSLHandshake.produce(ConnectionContext, SSLHandshake$HandshakeMessage) line: 437
    sun.security.ssl.Finished$T13FinishedConsumer.onConsumeFinished(ClientHandshakeContext, ByteBuffer) line: 1030
    sun.security.ssl.Finished$T13FinishedConsumer.consume(ConnectionContext, ByteBuffer) line: 893
    sun.security.ssl.SSLHandshake.consume(ConnectionContext, ByteBuffer) line: 393
    sun.security.ssl.HandshakeContext.dispatch(byte, ByteBuffer) line: 476
    sun.security.ssl.HandshakeContext.dispatch(byte, Plaintext) line: 447
    sun.security.ssl.TransportContext.dispatch(Plaintext) line: 199
    sun.security.ssl.SSLTransport.decode(TransportContext, ByteBuffer[], int, int, ByteBuffer[], int, int) line: 172
    sun.security.ssl.SSLSocketImpl.decode(ByteBuffer) line: 1506
    sun.security.ssl.SSLSocketImpl.readHandshakeRecord() line: 1421
    sun.security.ssl.SSLSocketImpl.startHandshake(boolean) line: 455
    sun.security.ssl.SSLSocketImpl.startHandshake() line: 426
    sun.net.www.protocol.https.HttpsClient.afterConnect() line: 482
    sun.net.www.protocol.https.AbstractDelegateHttpsURLConnection.connect() line: 187
    sun.net.www.protocol.http.HttpURLConnection.getInputStream0() line: 1376
    sun.net.www.protocol.http.HttpURLConnection.getInputStream() line: 1301
    java.net.HttpURLConnection.getResponseCode() line: 493
    sun.net.www.protocol.https.HttpsURLConnectionImpl.getResponseCode() line: 307
    HttpsClient.processUrl(URL) line: 80
```

jdk.X509Certificate

- Records details of each X509 certificate generated

```
jdk.X509Certificate {
  startTime = 12:06:02.855 (2025-01-14)
  algorithm = "SHA256withRSA"
  serialNumber = "00:dl:ll:70:19:90:87:ll:16:09:fd:16:85:af:b6:74:34"
  subject = "CN=*.google.com"
  issuer = "CN=WR2, O=Google Trust Services, C=US"
  keyType = "EC"
  keyLength = 256
  certificateId = 1533204685
  validFrom = 08:36:18.000 (2024-12-09)
  validUntil = 08:36:17.000 (2025-03-03)
  eventThread = "main" (javaThreadId = 3)
  stackTrace = [
    sun.security.jca.JCAUtil.tryCommitCertEvent(Certificate) line: 128
    java.security.cert.CertificateFactory.generateCertificate(InputStream) line: 356
    sun.security.ssl.CertificateMessage$T13CertificateConsumer.checkServerCerts(ClientHandshakeContext, List) line: 1286
    sun.security.ssl.CertificateMessage$T13CertificateConsumer.onConsumeCertificate(ClientHandshakeContext, CertificateMessage$T13
    sun.security.ssl.CertificateMessage$T13CertificateConsumer.consume(ConnectionContext, ByteBuffer) line: 1146
    sun.security.ssl.SSLHandshake.consume(ConnectionContext, ByteBuffer) line: 393
    sun.security.ssl.HandshakeContext.dispatch(byte, ByteBuffer) line: 476
    sun.security.ssl.HandshakeContext.dispatch(byte, Plaintext) line: 447
    sun.security.ssl.TransportContext.dispatch(Plaintext) line: 199
    sun.security.ssl.SSLTransport.decode(TransportContext, ByteBuffer[], int, int, ByteBuffer[], int, int) line: 172
    sun.security.ssl.SSLSocketImpl.decode(ByteBuffer) line: 1506
    sun.security.ssl.SSLSocketImpl.readHandshakeRecord() line: 1421
    sun.security.ssl.SSLSocketImpl.startHandshake(boolean) line: 455
    sun.security.ssl.SSLSocketImpl.startHandshake() line: 426
    sun.net.www.protocol.https.HttpsClient.afterConnect() line: 482
    sun.net.www.protocol.https.AbstractDelegateHttpsURLConnection.connect() line: 187
    sun.net.www.protocol.http.HttpURLConnection.getInputStream0() line: 1376
    sun.net.www.protocol.http.HttpURLConnection.getInputStream() line: 1301
    java.net.HttpURLConnection.getResponseCode() line: 493
    sun.net.www.protocol.https.HttpsURLConnectionImpl.getResponseCode() line: 307
```

jdk.X509Validation

- Records details relating to chain of trust for successful CertPath operations

```
jdk.X509Validation {
  startTime = 12:06:02.858 (2025-01-14)
  certificateId = 657172038
  certificatePosition = 1
  validationCounter = 2
  eventThread = "main" (javaThreadId = 3)
  stackTrace = [
    sun.security.provider.certpath.PKIXCertPathValidator.validate(TrustAnchor, PKIX$ValidatorParams) line: 242
    sun.security.provider.certpath.PKIXCertPathValidator.validate(PKIX$ValidatorParams) line: 144
    sun.security.provider.certpath.PKIXCertPathValidator.engineValidate(CertPath, CertPathParameters) line: 83
    java.security.cert.CertPathValidator.validate(CertPath, CertPathParameters) line: 307
    sun.security.validator.PKIXValidator.doValidate(X509Certificate[], PKIXBuilderParameters) line: 312
    sun.security.validator.PKIXValidator.engineValidate(X509Certificate[], Collection, List, AlgorithmConstraints, Object) line: 255
    sun.security.validator.Validator.validate(X509Certificate[], Collection, List, AlgorithmConstraints, Object) line: 256
    sun.security.ssl.X509TrustManagerImpl.checkTrusted(X509Certificate[], String, Socket, boolean) line: 230
    sun.security.ssl.X509TrustManagerImpl.checkServerTrusted(X509Certificate[], String, Socket) line: 132
    sun.security.ssl.CertificateMessage$T13CertificateConsumer.checkServerCerts(ClientHandshakeContext, List) line: 1310
    sun.security.ssl.CertificateMessage$T13CertificateConsumer.onConsumeCertificate(ClientHandshakeContext, CertificateMessage$T13CertificateMessage)
    sun.security.ssl.CertificateMessage$T13CertificateConsumer.consume(ConnectionContext, ByteBuffer) line: 1146
    sun.security.ssl.SSLHandshake.consume(ConnectionContext, ByteBuffer) line: 393
    sun.security.ssl.HandshakeContext.dispatch(byte, ByteBuffer) line: 476
    sun.security.ssl.HandshakeContext.dispatch(byte, Plaintext) line: 447
    sun.security.ssl.TransportContext.dispatch(Plaintext) line: 199
    sun.security.ssl.SSLTransport.decode(TransportContext, ByteBuffer[], int, int, ByteBuffer[], int, int) line: 172
    sun.security.ssl.SSLSocketImpl.decode(ByteBuffer) line: 1506
    sun.security.ssl.SSLSocketImpl.readHandshakeRecord() line: 1421
    sun.security.ssl.SSLSocketImpl.startHandshake(boolean) line: 455
    sun.security.ssl.SSLSocketImpl.startHandshake() line: 426
    sun.net.www.protocol.https.HttpsClient.afterConnect() line: 482
```


jdk.InitialSecurityProperty

- Records details of all security properties at JVM initialize phase

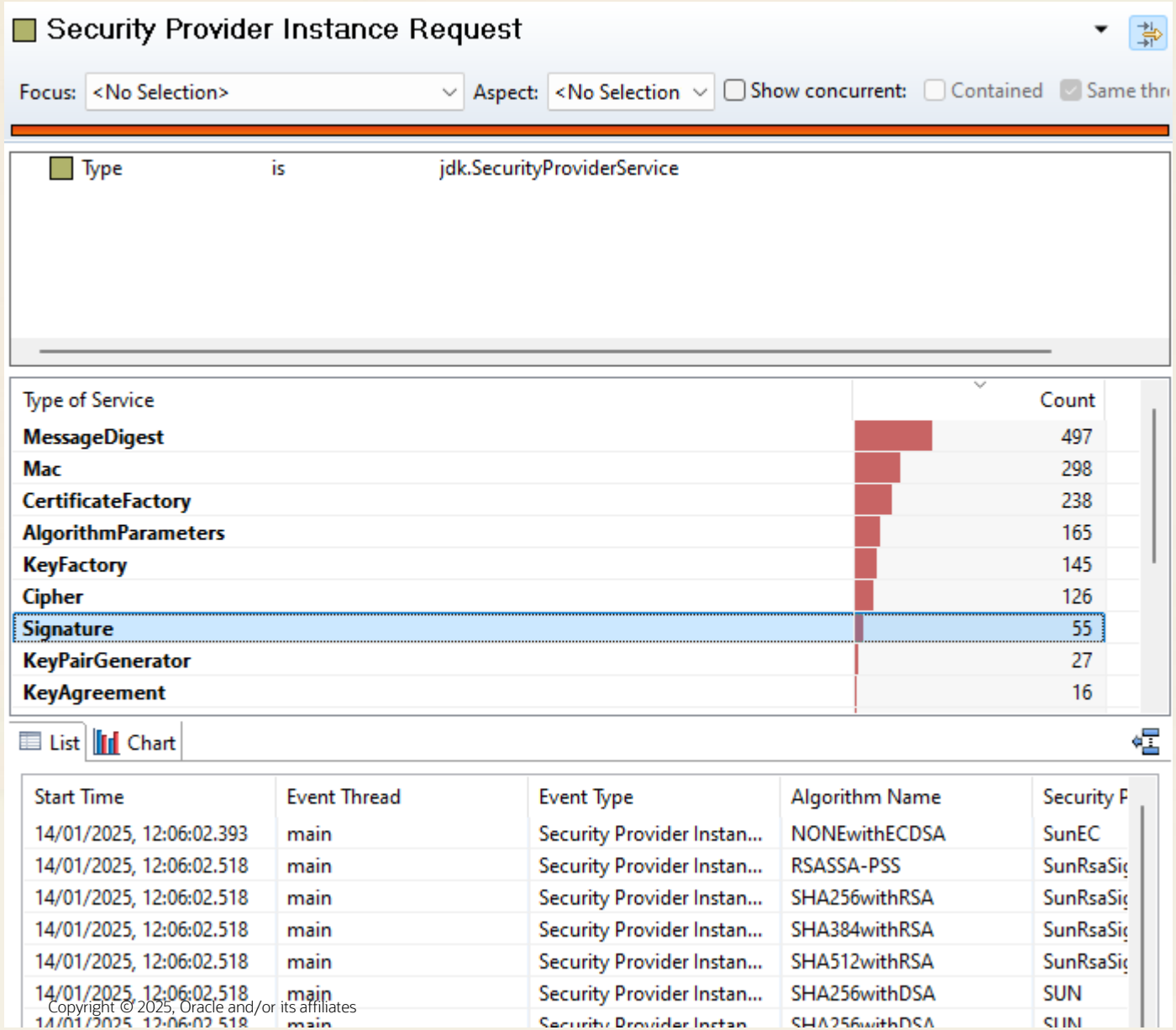
```
jdk.InitialSecurityProperty {
  startTime = 12:06:02.321 (2025-01-14)
  key = "http.auth.digest.disabledAlgorithms"
  value = "MD5, SHA-1"
}

jdk.InitialSecurityProperty {
  startTime = 12:06:02.321 (2025-01-14)
  key = "jdk.security.legacyAlgorithms"
  value = "SHA1, RSA keySize < 2048, DSA keySize < 2048, DES, DESede, MD5, RC2, ARCFOUR"
}

jdk.InitialSecurityProperty {
  startTime = 12:06:02.321 (2025-01-14)
  key = "securerandom.source"
  value = "file:/dev/random"
}
```

jdk.SecurityProviderService

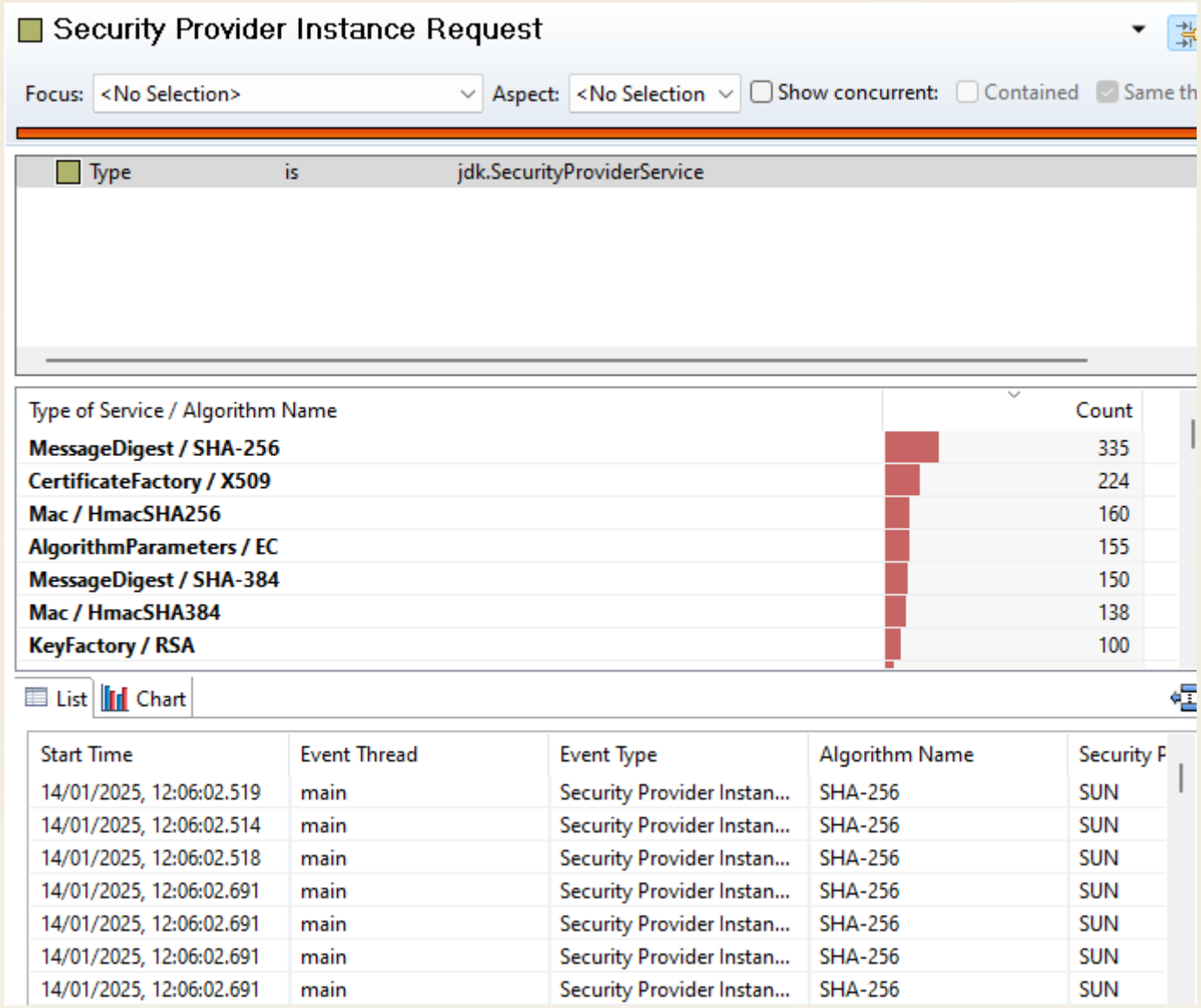
- Records details relating to every crypto request made to the JCE framework

```
jdk.SecurityProviderService {
  startTime = 12:06:06.983 (2025-01-14)
  type = "MessageDigest"
  algorithm = "SHA-256"
  provider = "SUN"
  eventThread = "main" (javaThreadId = 3)
  stackTrace = [
    sun.security.jca.ProviderList.getService(String, String) line: 378
    sun.security.jca.GetInstance.getInstance(String, Class, String) line: 145
    java.security.MessageDigest.getInstance(String) line: 185
    com.sun.crypto.provider.HmacCore.<init>(String, int) line: 64
    com.sun.crypto.provider.HmacCore$HmacSHA256.<init>() line: 275
    jdk.internal.reflect.DirectConstructorHandleAccessor.newInstance(Object[]) line: 62
    java.lang.reflect.Constructor.newInstanceWithCaller(Object[], boolean, Class) line: 499
    java.lang.reflect.Constructor.newInstance(Object[]) line: 483
    java.security.Provider$Service.newInstanceOf() line: 1768
    java.security.Provider$Service.newInstanceUtil(Class, Object) line: 1775
    java.security.Provider$Service.newInstance(Object) line: 1750
    javax.crypto.Mac.chooseFirstProvider() line: 316
    javax.crypto.Mac.getMacLength() line: 412
    sun.security.ssl.HKDF.<init>(String) line: 65
    sun.security.ssl.NewSessionTicket.derivePreSharedKey(CipherSuite$HashAlg, SecretKey, byte[]) line: 289
    sun.security.ssl.NewSessionTicket$T13NewSessionTicketConsumer.consume(ConnectionContext, ByteBuffer) line: 615
    sun.security.ssl.SSLHandshake.consume(ConnectionContext, ByteBuffer) line: 393
    sun.security.ssl.PostHandshakeContext.dispatch(byte, ByteBuffer) line: 82
    sun.security.ssl.HandshakeContext.dispatch(byte, Plaintext) line: 447
    sun.security.ssl.TransportContext.dispatch(Plaintext) line: 199
    sun.security.ssl.SSLTransport.decode(TransportContext, ByteBuffer[], int, int, ByteBuffer[], int, int) line: 172
    sun.security.ssl.SSLSocketImpl.decode(ByteBuffer) line: 1509
    sun.security.ssl.SSLSocketImpl.readApplicationRecord(ByteBuffer) line: 1480
    sun.security.ssl.SSLSocketImpl$AppInputStream.read(byte[], int, int) line: 1068
```

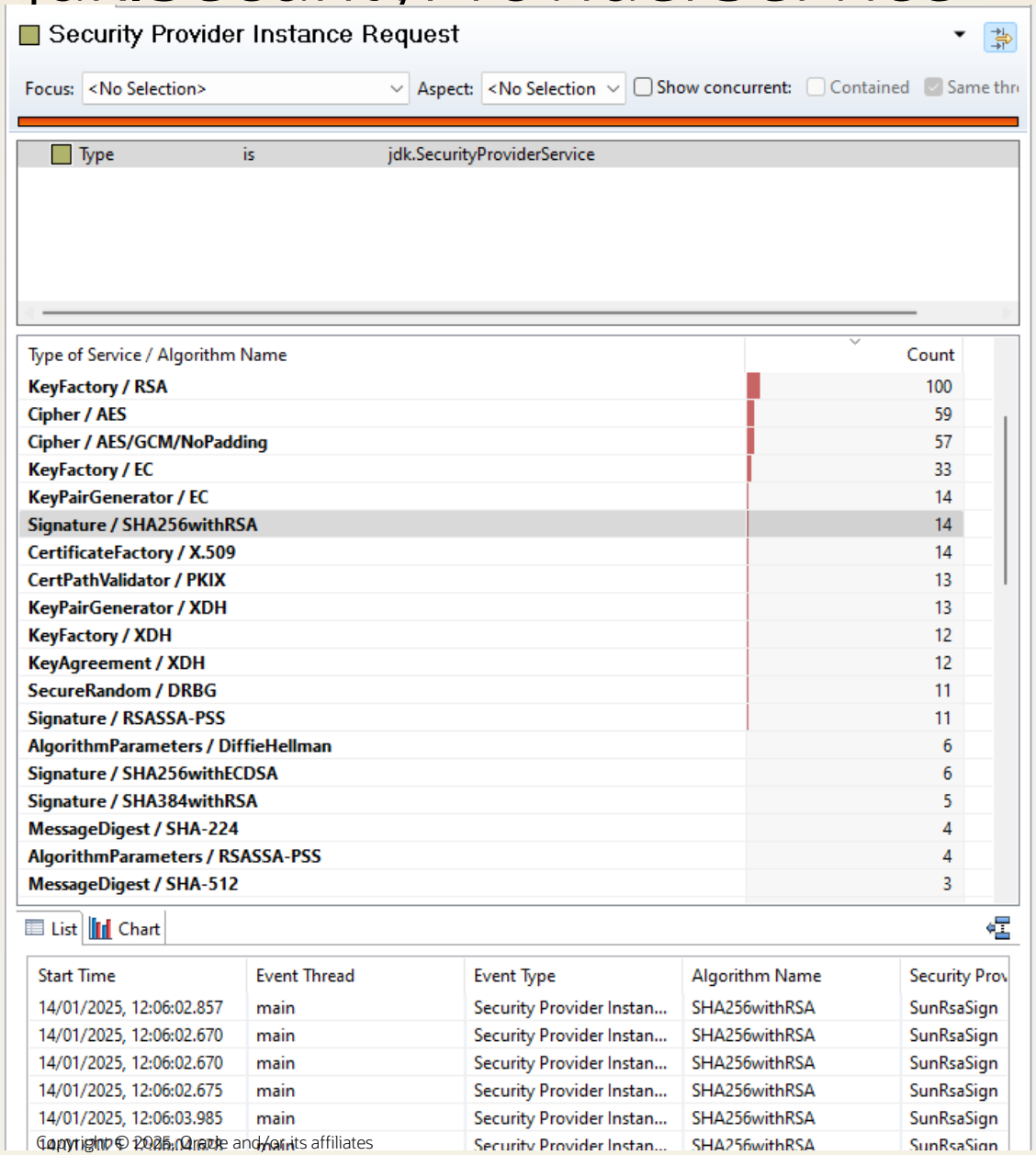
jdk.SecurityProviderService

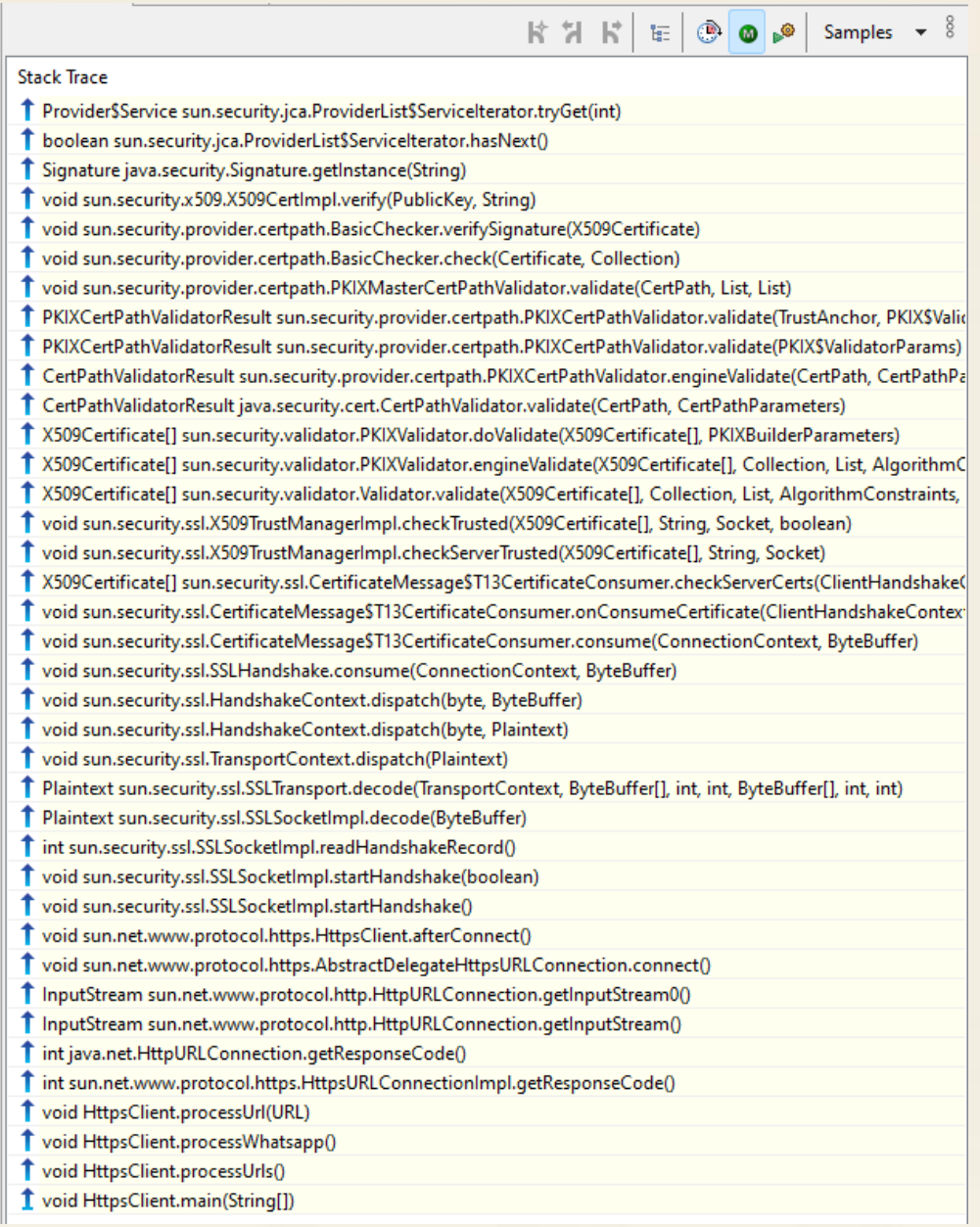


jdk.SecurityProviderService (group by)



jdk.SecurityProviderService





JFR Event use cases

- https://www.java.com/en/jre-jdk-cryptoroadmap.html

Released Changes

Release Date	Release(s) Affected	Algorithm/Protocol	Action	Additional Information	Change Log
2024-10-15	17.0.13+10, 11.0.25+9, 8u431 b10	TLS	Disabled TLS_ECDH Cipher Suites	Disabling TLS_ECDH Cipher Suites	• 2024-10-15 Released. • 2024-04-30 Announced.
2024-10-15	23.0.1+11, 21.0.5+9, 17.0.13+10, 11.0.25+9, 8u431 b10	TLS	Distrust TLS server certificates anchored by Entrust root certificates and issued after Nov 11, 2024	Distrusting TLS server certificates anchored by Entrust root certificates and issued after Nov 11, 2024	• 2024-10-15 Released. • 2024-09-30 Changed distrust date from after Oct to after Nov 11. • 2024-09-04 Announced.
2024-07-16	17.0.12+8, 11.0.24+7	DTLS 1.0	Disabled DTLS 1.0	Disabling DTLS 1.0	• 2024-07-16 Released. • 2024-01-22 Announced.

JFR Event use cases

- https://www.java.com/en/jre-jdk-cryptoroadmap.html

Released Changes

Release Date	Release(s) Affected	Algorithm/Protocol	Action	Additional Information	Change Log
2024-10-15	17.0.13+10, 11.0.25+9, 8u431 b10	TLS	Disabled TLS_ECDH Cipher Suites	Disabling TLS_ECDH Cipher Suites	• 2024-10-15 Released.
2024-10-15	23.0.1+11, 21.0.5+9, 17.0.13+10, 11.0.25+9, 8u431 b10	TLS	Distrust TLS server certificates anchored by Entrust root certificates and issued after Nov 11, 2024	Distrusting TLS server certificates anchored by Entrust root certificates and issued after Nov 11, 2024	• 2024-10-15 Released. • 2024-09-30 Changed distrust date from after Oct to after Nov 11. • 2024-09-04 Announced.
2024-07-16	17.0.12+8, 11.0.24+7	DTLS 1.0	Disabled DTLS 1.0	Disabling DTLS 1.0	• 2024-07-16 Released. • 2024-01-22 Announced.

jdk.TLSHandshake

JFR Event use cases

- https://www.java.com/en/jre-jdk-cryptoroadmap.html

Released Changes

Release Date	Release(s) Affected	Algorithm/Protocol	Action	Additional Information	Change Log
2024-10-15	17.0.13+10, 11.0.25+9, 8u431 b10	TLS	Disabled TLS_ECDH Cipher Suites	Disabling TLS_ECDH Cipher Suites	• 2024-10-15 Released.
2024-10-15	23.0.1+11, 21.0.5+9, 17.0.13+10, 11.0.25+9, 8u431 b10	TLS	Distrust TLS server certificates anchored by Entrust root certificates and issued after Nov 11, 2024	Distrusting TLS server certificates anchored by Entrust root certificates and issued after Nov 11, 2024	• 2024-10-15 Released. • 2024-09-30 Changed distrust date from after Nov 11. • 2024-09-04 Announced.
2024-07-16	17.0.12+8, 11.0.24+7	DTLS 1.0	Disabled DTLS 1.0	Disabling DTLS 1.0	• 2024-07-16 Released. • 2024-01-22 Announced.

jdk.TLSHandshake

jdk.X509Certificate

JFR Event use cases

- https://www.java.com/en/jre-jdk-cryptoroadmap.html

Released Changes

Release Date	Release(s) Affected	Algorithm/Protocol	Action	Additional Information	Change Log
2024-10-15	17.0.13+10, 11.0.25+9, 8u431 b10	TLS	Disabled TLS_ECDH Cipher Suites	Disabling TLS_ECDH Cipher Suites	• 2024-10-15 Released.
2024-10-15	23.0.1+11, 21.0.5+9, 17.0.13+10, 11.0.25+9, 8u431 b10	TLS	Distrust TLS server certificates anchored by Entrust root certificates and issued after Nov 11, 2024	Distrusting TLS server certificates anchored by Entrust root certificates and issued after Nov 11, 2024	• 2024-10-15 Released. • 2024-09-30 Changed distrust date from after Nov 11. • 2024-09-04 Announced.
2024-07-16	17.0.12+8, 11.0.24+7	DTLS 1.0	Disabled DTLS 1.0	Disabling DTLS 1.0	• 2024-07-16 Released.

jdk.TLSHandshake

jdk.X509Certificate

jdk.TLSHandshake

More complex case study: Upgrading the default PKCS12 MAC algorithm

2022-10-18	11.0.17+10, 8u351 b10	SHA-2	Upgraded the default PKCS12 MAC algorithm	Upgrading the default PKCS12 MAC algorithm

Upgrading the default PKCS12 MAC algorithm

Note that this change has already been made in JDK 16 and later releases.

The default algorithms used to protect the integrity of a PKCS12 keystore will be upgraded to a stronger algorithm.

This change is targeted to the JDK 7, 8, and 11 update releases. To test the change on those releases, you must be using the July 2021 CPU JDK releases (7u311, 8u301, 11.0.12) or later. There is no need to test on other JDK supported releases since the change is in 16 and later. To test the changes before they take affect, edit the `java.security` file and uncomment (remove the leading "#") and modify the values of the following security properties:

```
keystore.pkcs12.macAlgorithm = HmacPBESHA256
```

Since the algorithms are stronger, the iteration count security properties can also be adjusted according to the recommended iteration count as described in NIST Special Publication 800-63B "5.1.1.2 Memorized Secret Verifiers":

```
keystore.pkcs12.macIterationCount = 10000
```

These properties will be used by the PKCS12 `KeyStore` implementation during the creation of a new keystore.

More complex case study: Upgrading the default PKCS12 MAC algorithm

2022-10-18	11.0.17+10, 8u351 b10	SHA-2	Upgraded the default PKCS12 MAC algorithm	Upgrading the default PKCS12 MAC algorithm

Upgrading the default PKCS12 MAC algorithm

Note that this change has already been made in JDK 16 and later releases.

The default algorithms used to protect the integrity of a PKCS12 keystore will be upgraded to a stronger algorithm.

This change is targeted to the JDK 7, 8, and 11 update releases. To test the change on those releases, you must be using the July 2021 CPU JDK releases (7u311, 8u301, 11.0.12) or later. There is no need to test on other JDK supported releases since the change is in 16 and later. To test the changes before they take affect, edit the `java.security` file and uncomment (remove the leading "#") and modify the values of the following security properties:

```
keystore.pkcs12.macAlgorithm = HmacPBESHA256
```

Since the algorithms are stronger, the iteration count security properties can also be adjusted according to the recommended iteration count as described in NIST Special Publication 800-63B "5.1.1.2 Memorized Secret Verifiers":

```
keystore.pkcs12.macIterationCount = 10000
```

These properties will be used by the PKCS12 `KeyStore` implementation during the creation of a new keystore.

jdk.SecurityProviderService

PKCS12 MAC algorithm analysis

```
jdk.SecurityProviderService {
  startTime = 14:36:50.950 (2024-11-26)
  type = "Mac"
  algorithm = "HmacPBESHA1"
  provider = "SunJCE"
  eventThread = "main" (javaThreadId = 1)
  stackTrace = [
    java.security.Provider.getService(String, String) line: 1323
    java.security.Provider$Service.newInstance(Object) line: 1890
    javax.crypto.Mac.chooseProvider(Key, AlgorithmParameterSpec) line: 365
    javax.crypto.Mac.init(Key, AlgorithmParameterSpec) line: 465
    sun.security.pkcs12.PKCS12KeyStore.lambda$engineLoad$2(Mac, PBEParameterSpec, byte[], int, MacData,
    sun.security.pkcs12.PKCS12KeyStore$RetryWithZero.run(PKCS12KeyStore$RetryWithZero, char[]) line: 25
    sun.security.pkcs12.PKCS12KeyStore.engineLoad(InputStream, char[]) line: 2133
    sun.security.util.KeyStoreDelegator.engineLoad(InputStream, char[]) line: 222
    java.security.KeyStore.load(InputStream, char[]) line: 1479
    LoadPKCS12.main(String[]) line: 17
  ]
```

PKCS12 MAC algorithm analysis

```
jdk.SecurityProviderService {
  startTime = 14:36:50.950 (2024-11-26)
  type = "Mac"
  algorithm = "HmacPBESHA1"
  provider = "SunJCE"
  eventThread = "main" (javaThreadId = 1)
  stackTrace = [
    java.security.Provider.getService(String, String) line: 1323
    java.security.Provider$Service.newInstance(Object) line: 1890
    javax.crypto.Mac.chooseProvider(Key, AlgorithmParameterSpec) line: 365
    javax.crypto.Mac.init(Key, AlgorithmParameterSpec) line: 465
    sun.security.pkcs12.PKCS12KeyStore.lambda$engineLoad$2(Mac, PBEParameterSpec, byte[], int, MacData,
    sun.security.pkcs12.PKCS12KeyStore$RetryWithZero.run(PKCS12KeyStore$RetryWithZero, char[]) line: 25
    sun.security.pkcs12.PKCS12KeyStore.engineLoad(InputStream, char[]) line: 2133
    sun.security.util.KeyStoreDelegator.engineLoad(InputStream, char[]) line: 222
    java.security.KeyStore.load(InputStream, char[]) line: 1479
    LoadPKCS12.main(String[]) line: 17
  ]
```

PKCS12 MAC algorithm analysis

```
jdk.SecurityProviderService {
  startTime = 14:36:50.950 (2024-11-26)
  type = "Mac"
  algorithm = "HmacPBESHA1"
  provider = "SunJCE"
  eventThread = "main" (javaThreadId = 1)
  stackTrace = [
    java.security.Provider.getService(String, String) line: 1323
    java.security.Provider$Service.newInstance(Object) line: 1890
    javax.crypto.Mac.chooseProvider(Key, AlgorithmParameterSpec) line: 365
    javax.crypto.Mac.init(Key, AlgorithmParameterSpec) line: 465
    sun.security.pkcs12.PKCS12KeyStore.lambda$engineLoad$2(Mac, PBEParameterSpec, byte[], int, MacData,
    sun.security.pkcs12.PKCS12KeyStore$RetryWithZero.run(PKCS12KeyStore$RetryWithZero, char[]) line: 25
    sun.security.pkcs12.PKCS12KeyStore.engineLoad(InputStream, char[]) line: 2133
    sun.security.util.KeyStoreDelegator.engineLoad(InputStream, char[]) line: 222
    java.security.KeyStore.load(InputStream, char[]) line: 1473
    LoadPKCS12.main(String[]) line: 17
  ]
```

WEAK CRYPTO

PKCS12 MAC algorithm analysis

```
jdk.SecurityProviderService {
  startTime = 13:43:36.648 (2023-04-26)
  type = "Mac"
  algorithm = "HmacPBESHA256"
  provider = "SunJCE"
  eventThread = "main" (javaThreadId = 1)
  stackTrace = [
    java.security.Provider.getService(String, String) line: 1323
    java.security.Provider$Service.newInstance(Object) line: 1890
    javax.crypto.Mac.chooseProvider(Key, AlgorithmParameterSpec) line: 365
    javax.crypto.Mac.init(Key, AlgorithmParameterSpec) line: 465
    sun.security.pkcs12.PKCS12KeyStore.lambda$engineLoad$2(Mac, PBEParameterSpec, byte[], int, MacData,
    sun.security.pkcs12.PKCS12KeyStore$RetryWithZero.run(PKCS12KeyStore$RetryWithZero  char[]) line: 25
    sun.security.pkcs12.PKCS12KeyStore.engineLoad(InputStream, char[]) line: 2133
    sun.security.util.KeyStoreDelegator.engineLoad(InputStream, char[]) line: 222
    java.security.KeyStore.load(InputStream, char[]) line: 1479
    LoadPKCS12.main(String[]) line: 17
  ]
```

PKCS12 MAC algorithm analysis

```
jdk.SecurityProviderService {
  startTime = 13:43:36.648 (2023-04-26)
  type = "Mac"
  algorithm = "HmacPBESHA256"
  provider = "SunJCE"
  eventThread = "main" (javaThreadId = 1)
  stackTrace = [
    java.security.Provider.getService(String, String) line: 1323
    java.security.Provider$Service.newInstance(Object) line: 1890
    javax.crypto.Mac.chooseProvider(Key, AlgorithmParameterSpec) line: 365
    javax.crypto.Mac.init(Key, AlgorithmParameterSpec) line: 465
    sun.security.pkcs12.PKCS12KeyStore.lambda$engineLoad$2(Mac, PBEParameterSpec, byte[], int, MacData,
    sun.security.pkcs12.PKCS12KeyStore$RetryWithZero.run(PKCS12KeyStore$RetryWithZero  char[]) line: 25
    sun.security.pkcs12.PKCS12KeyStore.engineLoad(InputStream, char[]) line: 2133
    sun.security.util.KeyStoreDelegator.engineLoad(InputStream, char[]) line: 222
    java.security.KeyStore.load(InputStream, char[]) line: 1479
    LoadPKCS12.main(String[]) line: 17
  ]
```

STRONGER CRYPTO

Oracle Java Management Service (JMS)

- Observe and manage JDK deployments
- Crypto roadmap impact analysis
- Help keep your applications secure by identifying weak cryptography usage
- Analysis using JFR events to detect if any of the managed Java instances need modification due to changes on the JDK Cryptographic Roadmap

Q & A

- Thank you
- security-dev@openjdk.org