Installation von Smart Software Manager Satellite (SSMS) 5.1.0 fehlschlägt im KVM-basierten Kernel

Inhalt

Einführung

In diesem Dokument wird die Lösung des Problems beschrieben, das auftritt, wenn die Smart Software Manager Satellite (SSMS) 5.1.0-Installation im Kernel mit KVM-Technologie (Keyboard/Video/Mouse), der die Cisco Cloud Service Platform umfasst, fehlschlägt.

Problem

Die Installation wird über die Konsole abgeschlossen, und der Zugriff auf die Benutzeroberfläche ist möglich.

Beim Setup des CSSM-Registrierungsprozesses wird festgestellt, dass die Registrierung fehlschlägt, während die Netzwerkregistrierung sowie die manuelle Registrierung durchgeführt werden. Die Tomcat-Version ist im KVM-basierten System validiert, Kernel und Java Virtual Machine (JVM). Beachten Sie, dass JVM 1.8.0_102-b14 und Kernel 3.10.0-514.el7 ausführt. Vergleichen Sie die ESXI-basierte Konfiguration, bei der der Kernel 3.10.0-862.14.4.el7 und JVM 1.8.0_191-b12 ausgeführt werden.

```
[root@satellite bin]# ./version.sh
Using CATALINA_BASE: /opt/tc
Using CATALINA_HOME: /opt/tc
Using CATALINA_TMPDIR: /opt/tc/temp
Using JRE_HOME: /
Using CLASSPATH: /opt/tc/bin/bootstrap.jar:/opt/tc/bin/tomcat-juli.jar
Using CATALINA_PID: /opt/tomcat/temp/tomcat.pid
Server version: Apache Tomcat/9.0.1
Server built: Sep 27 2017 17:31:52 UTC
Server number: 9.0.1.0
OS Name: Linux
OS Version: 3.10.0-514.el7.x86_64
Architecture: amd64
JVM Version: 1.8.0_102-b14
JVM Vendor: Oracle Corporation
```

Komponenten

Plattform: KVM-basierter Kernel

Software: Classic 5.1 ISO-Image

Lösung

Schritt 1: Navigieren Sie zu **cd/opt/tomcat/logs/**.

Schritt 2: Öffnen Sie **catalina.out-**Protokolle und suchen Sie die Ausnahme, die beim Registrierungsprozess bei CSSM auftritt.

IAIK-Provider IAIK-JCE ist eine Java-Kryptographieerweiterung, die über eine Reihe von APIs verfügt und kryptografische Funktionen implementieren kann. Es dient der Unterstützung zusätzlicher Sicherheitsfunktionen für das JDK. Das LCS-Modul kann aufgrund der Nichtverfügbarkeit der IAIK Jar-Datei kein Schlüsselpaar für die CSR-Anforderungsdatei generieren.

```
2019-05-15 20:35:01,604 [http-nio-8080-exec-9] INFO controller.LindosController - Invoked GET
/lcsSetupStatus
2019-05-15 20:35:01,606 [http-nio-8080-exec-9] INFO controller.LindosController - LCS Setup
Status = 0
2019-05-15 23:53:12,226 [http-nio-8080-exec-10] INFO controller.LindosController - Invoked GET
/lcsSetupStatus
2019-05-15 23:53:12,230 [http-nio-8080-exec-10] INFO controller.LindosController - LCS Setup
Status = 0
2019-05-15 23:53:12,241 [http-nio-8080-exec-1] INFO controller.LindosController - Invoked
/lcsSetup
2019-05-15 23:53:12,243 [http-nio-8080-exec-1] DEBUG controller.LindosController - Setup Status
= 0 (0=empty, 1=key/CSR generated, 2=Signer certs installed)
2019-05-15 23:53:12,243 [http-nio-8080-exec-1] DEBUG controller.LindosController - First time
setup invoked (ID element not present in JSON). CN=5fc62a80-59a0-0137-54ab-023a01ab3207
2019-05-15 23:53:12,243 [http-nio-8080-exec-1] DEBUG domain.LcsSignerSetup - In LcsSignerSetup
2019-05-15 23:53:12,244 [http-nio-8080-exec-1] DEBUG domain.LcsSignerSetup - Generating Key
Pair...
2019-05-15 23:53:12,244 [http-nio-8080-exec-1] ERROR error.RestResponseEntityExceptionHandler -
java.security.NoSuchProviderException: no such provider: IAIK
com.cisco.ias.lindos.data.domain.LcsSetupException: java.security.NoSuchProviderException: no
such provider: IAIK
at com.cisco.ias.lindos.data.domain.LcsSignerSetup.<init>(LcsSignerSetup.java:50)
at com.cisco.ias.lindos.web.controller.LindosController.setupLcs(LindosController.java:126)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at
org.springframework.web.method.support.InvocableHandlerMethod.invoke(InvocableHandlerMethod.java
:215)
at
org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerM
ethod.java:132)
at
org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHan
dle(ServletInvocableHandlerMethod.java:104)
at
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandleM
ethod(RequestMappingHandlerAdapter.java:749)
at
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInterna
l(RequestMappingHandlerAdapter.java:690)
at
```

```
org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapter.handle(AbstractHandlerMe
thodAdapter.java:83)
at org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:945)
at org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:876)
at org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:961)
at org.springframework.web.servlet.FrameworkServlet.doPost(FrameworkServlet.java:863)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:660)
at org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:837)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:741)
at
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:231
)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166)
at org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53)
at
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:193
)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166)
at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:199)
at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:96)
at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:140)
at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:81)
at org.apache.catalina.valves.AbstractAccessLogValve.invoke(AbstractAccessLogValve.java:651)
at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:87)
at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:342)
at org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:500)
at org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.java:66)
at org.apache.coyote.AbstractProtocol$ConnectionHandler.process(AbstractProtocol.java:754)
at org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1376)
at org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:49)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
at java.lang.Thread.run(Thread.java:745)
2019-05-15 23:53:12,254 [http-nio-8080-exec-2] INFO controller.LindosController - Invoked GET
/lcsSetupStatus
2019-05-15 23:53:12,256 [http-nio-8080-exec-2] INFO controller.LindosController - LCS Setup
Status = 0
```

Schritt 3: Platzieren Sie den erforderlichen Sicherheitsanbieter in classpath; **cp /opt/tomcat/webapps/Lindos/WEB-INF/lib/iaik_jce-5.1.jar /usr/lib/jvm/java/jre/lib/ext/**l.

Schritt 4: Stellen Sie sicher, dass das Glas von anderen Modulen lesbar ist: **chmod o+r /usr/lib/jvm/java/jre/lib/ext/iaik_jce-5.1.jar**.

Schritt 5: Speichern Sie den **java.security**-Dateipfad in eine temporäre Variable; **java_security=/usr/lib/jvm/java/jre/lib/security/java.security**.

Schritt 6: Erhöhen Sie die Priorität vorhandener Anbieter nach; **perl -pi -e's/^security.provider.(\d+)/"security.provider". . ($1+1)/e' $java_security**.

Schritt 7: Fügen Sie IAIK als ersten Anbieter in der Liste ein (beachten Sie den umgekehrten Schrägstrich, der Zeilenumbrüchen entgeht); **sed -i '/security.provider.2/i **

security.provider.1=iaik.security.provider.IAIK' $java_security.

Schritt 8: Starten Sie tomcat auf Änderungen neu, um mit dem Befehl zu starten; **systemctl restart tomcat**.

Schritt 9: Registrieren Sie den Satelliten beim CSSM, und wenn die Registrierung in Satellit

abgeschlossen ist, kann die Benutzeroberfläche nicht neu gestartet werden.

Schritt 10: Falten Sie beide x509-Zertifikate für Transport Layer Security (TLS)-Verbindungen an den Ports 443 und 8443 ein, um das PEM-Format (Privacy Enhanced Email) zu erfüllen. **Falten Sie -w 64 /drbd/certs/rails_ssl.crt > /drbd/certs/rails_ssl_folded.crt & mv /drbd/certs/rails_ssl_folded.crt /drbd/certs/rails_ssl.crt**

fold -w 64 /drbd/certs/pi_ssl.crt > /drbd/certs/pi_ssl_folded.crt && mv /drbd/certs/pi_ssl_folded.crt /drbd/certs/pi_ssl.crt.

> **Hinweis**: Führen Sie diese Befehle nicht faltend und nicht in einer anderen Zeile aus, da sie das 64-kodierte PEM-Zertifikat beschädigen.

Schritt 11.Start nginx; **systemctl start nginx**.

> **Hinweis**: Wenn die Benutzeroberfläche nach einer Synchronisierung nicht angezeigt wird, liegt dies daran, dass diese Zertifikate aktualisiert/ersetzt werden. Aus diesem Grund müssen die Schritte 8 bis 10 wiederholt werden.

Wenn Sie diese Schritte ausführen, greifen Sie auf die Benutzeroberfläche zu, und Sie sehen, dass die Synchronisierung mit dem CSSM abgeschlossen wurde. Die endgültige Registrierung ist erfolgreich.

Sie können den Bestands- und Lizenzabschnitt der VA-Lizenz zuordnen. Sie können Smart Product Instanzen auf Satellite registrieren.