Secure Hash Algorithm (SHA) 256 für Customer Voice Portal (CVP)

Inhalt

Einführung

Dieses Dokument beschreibt das Verfahren zur Verwendung von SHA256 mit CVP.

Voraussetzungen

Anforderungen

Cisco empfiehlt, über Kenntnisse in folgenden Bereichen zu verfügen:

- CVP
- Zertifikate

Verwendete Komponenten

Die Informationen in diesem Dokument basieren auf CVP 10.5.

Die Informationen in diesem Dokument wurden von den Geräten in einer bestimmten Laborumgebung erstellt. Alle in diesem Dokument verwendeten Geräte haben mit einer leeren (Standard-)Konfiguration begonnen. Wenn Ihr Netzwerk in Betrieb ist, stellen Sie sicher, dass Sie die potenziellen Auswirkungen eines Befehls verstehen.

Hintergrundinformationen

Ab Januar 2016 haben alle Browser SHA1-signierte Zertifikate zurückgewiesen. Dadurch wurden die angeforderten Services nicht korrekt wiedergegeben, es sei denn, Sie wechseln von SHA1 zu SHA256.

Mit der jüngsten Entwicklung in Computational-Algorithmen und der explosiven Rechenleistung ist SHA1 Tag für Tag schwächer geworden. Dies führte zu einer fundamentalen Verschlechterung

der Kollision Widerstand der SHA1 und schließlich zum Niedergang.

Konfigurieren

Verfahren für den Zertifikataustausch zwischen der CVP Operations Console (OAMP):

OAMP

Schritt 1: OAMP CERT exportieren.

c:\Cisco\CVP\jre\bin\keytool.exe -export -v -keystore .keystore -storetype JCEKS -alias oamp_certificate -file oamp_security_76.cer

Schritt 2: Kopieren Sie das OAMP-Zertifikat auf den Callserver, und importieren Sie es.

c:\Cisco\CVP\jre\bin\keytool.exe -import -trustcacerts -keystore -keystore -storetype JCEKS -alias orm_oamp_certificate -file oamp_security_76.cer

On Call Server

Schritt 1: CALLSERVER CERT exportieren.

c:\Cisco\CVP\jre\bin\keytool.exe -export -v -keystore .ormkeystore -storetype JCEKS -alias orm_certificate -file orm_security_108.cer

Schritt 2: Kopieren Sie CALLSERVER CERT in OAMP und importieren Sie es.

c:\Cisco\CVP\jre\bin\keytool.exe -import -trustcacerts -keystore -keystore -storetype JCEKS -alias oamp_orm_certificate -file orm_security_108.cer

Schritt 3: Exportieren Sie das Formularzertifikat im Call Server-Keystore.

C:\Cisco\CVP\conf\security>c:\Cisco\CVP\jre\bin\keytool.exe -import -trustcacerts -keystore -keystore -storetype JCEKS -alias vxml_orm_certificate -file orm_security_108.cer

Überprüfen

Sie können überprüfen, ob die sichere Kommunikation zwischen den Komponenten hergestellt wurde. Navigieren Sie zu **OAMP-Seite > Device Management > <managed server> > Statistics**

Statistiken müssen angezeigt werden.

Sie können JConsole verwenden, um eine Verbindung herzustellen, wenn die Sicherheit ordnungsgemäß eingerichtet ist:

Schritt 1: **c:\Cisco\CVP\conf\orm_jmx.conf** auf OAMP sieht wie folgt aus:

```
javax.net.debug = all
com.sun.management.jmxremote.ssl.need.client.auth = false
com.sun.management.jmxremote.authenticate = false
```

```
com.sun.management.jmxremote.port = 2099
com.sun.management.jmxremote.ssl = true
javax.net.ssl.keyStore=C:\Cisco\CVP\conf\security\.ormkeystore
javax.net.ssl.keyStorePassword=<local security password>
```

Schritt 2: Öffnen Sie jconsole über den Befehl. Verwenden Sie den Befehl:

C:\Cisco\CVP\jre\bin>jconsole.exe - J-Djavax.net.ssl.trustStore=C:\Cisco\CVP\conf\security\.keystore -J-Djavax.net.ssl.trustStorePassword=<Sicherheitskennwort/Jconsole-Client> - J-Djavax.net.ssl.keyStore=C:\Cisco\CVP\conf\security\.keystore -J-Djavax. .ssl.keyStorePassword=<Sicherheitskennwort/jconsole client> -J-Djavax.net.ssl.keyStoreType=JCEKS -debug -J-Djavax.net.ssl.trustStoreType=JCEKS

Key in **<managed server ip>:<secure jmx port eg:2099>** in **Remote Process** field.

> **Hinweis**: JConsole muss ohne Aufforderung eine Verbindung herstellen, damit die Anwendung die sichere Methode umgeht.

Schritt 3: Wireshark, während die jconsole-Verbindung aufgerufen wird. Die Erfassung bietet Ihnen Einblicke in die ausgehandelten Details, während die Sicherheit im Griff ist.

Spuren in JMX

Die Implementierung von JMX verwendet java.util.logging, um Debug-Traces zu protokollieren. Viele dieser Ablaufverfolgungen betreffen interne nicht exponierte Klassen, aber sie können Ihnen helfen, zu verstehen, was mit der Anwendung geschieht.

Die JMX-Implementierung verfügt über zwei Gruppen von Loggern:

- **`javax.management.\*`**: alle Protokollierungen im Zusammenhang mit der JMX API
- **`javax.management.remote.\*`**: Protokollierungen speziell für die JMX Remote API

Eine ausführlichere Beschreibung von JMX Loggers finden Sie hier.

Sie können die JMX-Spuren auf zwei verschiedene Arten aktivieren:

- Statisch unter Verwendung einer **logging.properties-**Datei
- Dynamisch, mit einem JMXTracing MBean. In Java SE 6 können Sie dies für eine Anwendung tun, auch wenn der JMX-Anschluss in der Befehlszeile nicht aktiviert ist.

Verwenden einer logging.properties-Datei

Starten Sie die Anwendung mit den folgenden Flags:

```
java -Djava.util.logging.config.file=<logging.properties> ....
```

wobei **logging.properties** Spuren für JMX-Protokollierungen aktiviert:

```
handlers= java.util.logging.ConsoleHandler
.level=INFO

java.util.logging.FileHandler.pattern = %h/java%u.log
java.util.logging.FileHandler.limit = 50000
java.util.logging.FileHandler.count = 1
java.util.logging.FileHandler.formatter = java.util.logging.XMLFormatter

java.util.logging.ConsoleHandler.level = FINEST
java.util.logging.ConsoleHandler.formatter = java.util.logging.SimpleFormatter

// Use FINER or FINEST for javax.management.remote.level - FINEST is
// very verbose...
//
javax.management.level=FINEST
javax.management.remote.level=FINER
```