# Secure Hash Algorithm (SHA) 256 para el portal de la voz del cliente (CVP)

## Contenido



## Introducción

Este documento describe el procedimiento para utilizar SHA256 con el CVP.

## Prerrequisitos

### Requisitos

Cisco recomienda que tenga conocimiento sobre estos temas:

- CVP
- Certificados

### Componentes Utilizados

La información en este documento se basa en el CVP 10.5.

La información que contiene este documento se creó a partir de los dispositivos en un ambiente de laboratorio específico. Todos los dispositivos que se utilizan en este documento se pusieron en funcionamiento con una configuración verificada (predeterminada). Si su red está viva, asegúrese de que usted entienda el impacto potencial del comando any.

## Antecedentes

Comenzando enero de 2016 todos los navegadores rechazaron los certificados firmados SHA1. Esto no hizo los servicios pedidos correctamente, a menos que usted se traslade desde el SHA1 A SHA256.

Con el reciente desarrollo en los algoritmos de cómputo así como la capacidad de cómputo explosiva, el SHA1 ha crecido más débil día a día. Esto llevó a la resistencia fundamental de la

colisión de la degradación del SHA1 y del fallecimiento eventual.

# Configurar

Certifique el procedimiento del intercambio entre la consola de las operaciones del CVP (OAMP):

En OAMP

Paso 1. Exportación OAMP CERT.

**c:\Cisco\CVP\jre\bin\keytool.exe - exportación - v - keystore .keystore - storetype JCEKS - alias oamp_certificate - archivo oamp_security_76.cer**

Paso 2. Copie el certificado OAMP a Callserver e impórtelo.

**c:\Cisco\CVP\jre\bin\keytool.exe - importación - los trustcacerts - el keystore .keystore - el storetype JCEKS - alias orm_oamp_certificate - clasifíe oamp_security_76.cer**

En el servidor de la llamada

Paso 1. Exportación CALLSERVER CERT.

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

Paso 2. Copie CALLSERVER CERT a OAMP e impórtelo.

**c:\Cisco\CVP\jre\bin\keytool.exe - importación - los trustcacerts - el keystore .keystore - el storetype JCEKS - alias oamp_orm_certificate - clasifíe orm_security_108.cer**

Paso 3. Certificado del orm de la exportación en el keystore del servidor de la llamada.

**C:\Cisco\CVP\conf\security > c:\Cisco\CVP\jre\bin\keytool.exe - importación - los trustcacerts - el keystore .keystore - el storetype JCEKS - alias vxml_orm_certificate - clasifían orm_security_108.cer**

# Verificación

Usted puede validar si la comunicación segura se establece entre los componentes. Navegue a la **página > a la Administración de dispositivos OAMP > server> > las estadísticas <managed**

El Stats debe ser visualizado.

Usted puede utilizar JConsole para establecer una conexión si la Seguridad se pone correctamente:

El paso 1. **c:\Cisco\CVP\conf\orm _jmx.conf en** OAMP parece:

```
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>
```

Paso 2. Abra el jconsole del comando.  Use el comando:

**C:\Cisco\CVP\jre\bin >jconsole.exe - J-Djavax.net.ss l.trustStore= C:\Cisco\CVP\conf\security\ .keystore - Contraseña de seguridad del =<oamp de J-Djavax.net.ss l.trustStorePassword/client> del jconsole - J-Djavax.net.ss l.keyStore = C:\Cisco\CVP\conf\security\ .keystore - contraseña de seguridad del =<oamp de J-Djavax.net.ss l.keyStorePassword/client> del jconsole - J-Djavax.net.ss l.keyStoreType=JCEKS - debug - J-Djavax.net.ss l.trustStoreType =JCEKS**

Clave en el **ip> <managed del servidor: puerto eg:2099> del jmx del <secure** en el campo **de proceso remoto**.

> **Note**: JConsole debe conectar sin el prompt para que la aplicación desvíe el método seguro.

Paso 3. Wireshark mientras que se invoca la conexión del jconsole. La captura le da la penetración en los detalles negociados mientras que apretón de manos de la Seguridad.

## Trazas en JMX

La implementación de las aplicaciones java.util.logging JMX de registrar las trazas del debug. Muchas de estas trazas se refieren a las clases no expuestas internas, pero pueden ayudarle a entender qué está continuando con su aplicación.

La implementación JMX tiene dos conjuntos de los madereros:

- **`javax.management. \ *: todos los`** madereros relacionados con el JMX API
- **`javax.management.remote. \ *: madereros`** relacionados **`específicamente`** con el telecontrol API JMX

Usted puede encontrar una más descripción completa de los madereros JMX aquí.

Usted puede activar las trazas JMX en dos maneras diferentes:

- Estáticamente, con el uso de un **archivo logging.properties**
- Dinámicamente, con el uso de un JMXTracing MBean. En las Javas SE 6, usted puede hacer esto para una aplicación, incluso si el conector JMX no se habilita en la línea de comando.

## Utilice un archivo logging.properties

Comience su aplicación con estos indicadores:

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

donde **logging.properties activa las** trazas para los madereros JMX:

```
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
```