Bereitstellen und Beheben von Fehlern bei Autorisierungscodes - Optimierung des OAuth-Programms: Cisco Collaboration-Lösungen 12.0

Inhalt

Einleitung

In diesem Dokument wird beschrieben, wie der Fluss der Autorisierungscode-Zuschüsse auf Aktualisierungstoken basiert, um die Benutzerfreundlichkeit von Jabber auf verschiedenen Geräten zu verbessern, insbesondere bei Jabber on Mobile.

Voraussetzungen

Anforderungen

Cisco empfiehlt, dass Sie über Kenntnisse in folgenden Bereichen verfügen:

- Cisco Unified Communications Manager (CUCM) 12.0-Version
- Single Sign On (SSO)/SAML
- Cisco Jabber
- Microsoft ADFS
- Identitätsanbieter (IdP)

Weitere Informationen zu diesen Themen finden Sie unter:

- SAML SSO-Bereitstellungsleitfaden für Cisco Unified Communications
- Unified Communications Manager SAML SSO-Konfigurationsbeispiel:

- AD FS Version 2.0-Setup für SAML SSO-Konfigurationsbeispiel:

Verwendete Komponenten

Die Informationen in diesem Dokument basieren auf dieser Software:

- Microsoft ADFS (IdP)
- LDAP Active Directory
- Cisco Jabber-Client
- CUCM 12.0

Die Informationen in diesem Dokument beziehen sich auf Geräte in einer speziell eingerichteten Testumgebung. Alle Geräte, die in diesem Dokument benutzt wurden, begannen mit einer gelöschten (Nichterfüllungs) Konfiguration. Wenn Ihr Netzwerk in Betrieb ist, stellen Sie sicher, dass Sie die potenziellen Auswirkungen eines Befehls verstehen.

Hintergrundinformationen

Zum gegenwärtigen Zeitpunkt basiert der Jabber SSO-Fluss mit Infrastruktur auf Implicit Grant Flow, bei dem der CUCM Authz-Service die kurzlebigen Zugriffstoken zuweist.

Nach Ablauf des Tokens für den Zugriff leitet CUCM Jabber zur erneuten Authentifizierung an IDP um.

Dies führt zu einer schlechten Benutzererfahrung, insbesondere bei Jabber auf dem Mobilgerät, bei dem der Benutzer häufig aufgefordert wird, Anmeldeinformationen einzugeben.

Die Security Re-Architecture-Lösung bietet außerdem einen Fluss für Autorisierungscode (mit dem Ansatz "Refresh Tokens" (erweiterbar auf Endpunkte/andere Collaboration-Anwendungen)) für die Vereinheitlichung von Jabber- und Endpunkt-Login-Fluss für SSO- und Nicht-SSO-Szenarien.

Wichtigste Funktionen

- Der Fluss der Autorisierungscodes basiert auf einem Aktualisierungstoken (erweiterbar auf Endpunkte/andere Collaboration-Anwendungen), um die Benutzerfreundlichkeit von Jabber auf verschiedenen Geräten zu verbessern, insbesondere für Jabber on Mobile.
- Unterstützt selbstenthaltene signierte und verschlüsselte OAuth-Token, um verschiedenen Collaboration-Anwendungen die Validierung und Beantwortung von Clientressourcenanforderungen zu ermöglichen.
- Das implizite Grant-Flow-Modell wird beibehalten, was Abwärtskompatibilität ermöglicht. Dies ermöglicht auch einen nahtlosen Pfad für andere Clients (wie RTMT), die noch nicht in den Autorisierungs-Code-Grant-Fluss verschoben wurden.

Wichtige Überlegungen

- Implementierung, sodass der alte Jabber-Client mit dem neuen CUCM arbeiten kann (da er sowohl implizite Zuweisen als auch Autorisierungscode-Zuweisungen unterstützt). Der neue Jabber kann auch mit dem alten CUCM verwendet werden. Jabber kann bestimmen, ob

CUCM den Fluss von Autorisierungscode-Finanzhilfen unterstützt und nur, wenn es dieses Modell unterstützt, wechselt und verwendet implizite Zuschüsse.
- Der AuthZ-Dienst wird auf dem CUCM-Server ausgeführt.
- AuthZ unterstützt nur implizite Grant Flow. Das bedeutet, dass kein Aktualisierungstoken/Offline-Zugriffstoken vorhanden war. Jedes Mal, wenn der Client ein neues Zugriffstoken wollte, muss der Benutzer sich erneut mit der IDP authentifizieren.
- Zugriffstoken wurden nur ausgegeben, wenn Ihre Bereitstellung SSO aktiviert ist. Nicht-SSO-Bereitstellungen funktionierten in diesem Fall nicht, und Access Token wurden nicht auf allen Schnittstellen konsistent verwendet.
- Zugriffs-Token sind nicht eigenständig, sondern bleiben im Speicher des Servers erhalten, der sie ausgegeben hat. Wenn CUCM1 das Zugriffstoken ausgestellt hat, kann es nur von CUCM1 überprüft werden. Wenn der Client versucht, auf den Service auf CUCM2 zuzugreifen, muss CUCM2 dieses Token auf CUCM1 validieren. Netzwerkverzögerungen (Proxymodus)
- Die Benutzerfreundlichkeit mobiler Clients ist sehr schlecht, da der Benutzer Anmeldeinformationen auf einem alphanumerischen Tastenfeld erneut eingeben muss, wenn sich der Benutzer mit der IdP erneut authentifiziert (normalerweise von 1 Stunde bis 8 Stunden, abhängig von mehreren Faktoren).
- Clients, die über mehrere Schnittstellen mit mehreren Anwendungen kommunizieren, müssen mehrere Anmeldeinformationen/Blöcke verwalten. Keine nahtlose Unterstützung für dieselbe Benutzeranmeldung von zwei ähnlichen Clients. Benutzer A meldet sich beispielsweise von Jabber-Instanzen an, die auf zwei verschiedenen iPhones ausgeführt werden.
- AuthZ zur Unterstützung von SSO- und Nicht-SSO-Bereitstellungen.
- AuthZ unterstützt impliziten Grant Flow + Autorisierungscode Grant Flow. Da es **abwärtskompatibel** ist**,** können Clients wie **RTMT** so lange arbeiten, bis sie sich anpassen.
- Bei der Vergabe des Autorisierungscodes gibt AuthZ Zugriff auf Token und Aktualisierungstoken. Mit dem Aktualisierungstoken kann ein anderes Zugriffstoken abgerufen werden, ohne dass eine Authentifizierung erforderlich ist.
- Zugriffstoken sind eigenständig, signiert und verschlüsselt und verwenden den JWT-Standard (JSON Web Tokens) (RFC-konform).
- Signierungs- und Verschlüsselungsschlüssel sind im Cluster üblich. Jeder Server im Cluster kann das Zugriffstoken überprüfen. Es ist nicht erforderlich, den Speicher beizubehalten.
- Der auf CUCM 12.0 ausgeführte Dienst ist der zentrale Authentifizierungsserver im Cluster.
- Aktualisierungs-Token werden in Datenbank (DB) gespeichert. Bei Bedarf muss der Administrator in der Lage sein, die Lizenz zu widerrufen. Der Widerruf basiert auf der Benutzer-ID, der Benutzer-ID und der Client-ID.
- Mit Token für den signierten Zugriff können verschiedene Produkte Zugriffstoken validieren, ohne dass diese gespeichert werden müssen. Konfigurierbare Zugriffstoken und Aktualisierungstoken - Lebensdauer (standardmäßig 1 Stunde bzw. 60 Tage).
- Das JWT-Format ist auf Spark abgestimmt, was in Zukunft Synergien mit Spark Hybrid-Services ermöglicht.
- Unterstützung für denselben Benutzer meldet sich von zwei ähnlichen Geräten an. Beispiel: Benutzer A kann sich über Jabber-Instanzen anmelden, die auf zwei verschiedenen iPhones ausgeführt werden.

Elemente des Codes für die Autorisierung, Fluss der Zuschüsse

- Auth Z-Server
- Verschlüsselungsschlüssel
- Signaturschlüssel
- Aktualisierungstoken

Konfigurieren

Diese Funktion ist standardmäßig nicht aktiviert.

Schritt 1: Um dieses Feature zu aktivieren, navigieren Sie zu **System > Enterprise Parameters (System > Enterprise-Parameter)**.

Schritt 2: Legen Sie den Parameter **OAuth mit Refresh Login Flow** auf Enabled (Aktiviert) fest, wie im Bild gezeigt.

- Zugriffs-Token wird signiert und verschlüsselt. Signierungs- und Verschlüsselungsschlüssel sind für den Cluster üblich. Das bedeutet, dass jeder Knoten im Cluster das Zugriffstoken validieren kann.
- Das Zugriffstoken hat das JWT-Format (RFC 7519).
- Zugriffstoken verwenden den Enterprise-Parameter (OAuth Access Token Expiry-Timer), der sowohl für alte Token- als auch für neue Tokenformate gilt.
- Standardwert: 60 Minuten.
- Mindestwert: 1 Minute.
- Maximaler Wert: 1440 Minuten


```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjhkMGQ1MzI0LWY0ZjAtNGIwYi04MTFlLTRhNTlmZGI2YjcyMjpj
Mjc3MGM5N2JkYTlkMzRmZDA1YTdlYTFhZWQzZTU0Y2E4MGJkZDdlZTM1ZDk3MDNiNjBiNTQ5MTBiZDQ0ODRiIn0.eyJwcml2
YXRlIjoiZXlKaGJHY2lPaUprYVhJaUxDSmpkSGtpT2lKS1YxUWlMQ0psYm1NaU9pSkJNVEk0UTBKRExVaFRNalUySWl3aWEy
bGtJam9pT0dRd1pEVXpNalF0WmpSbU1DMDBZakJpTFRneE1XVXROR0UxT1daa1lqWmlOekl5T21Vd1ptUm1ZMk16WlRRMU5E
RTFOV0ZpTkRJek5tRTJOMlV4T0RCbU1qWmxZMkl3WXpJeE56SXlOREJtWlRFellXWXlOak14TkRkalpHVXpNR1l3TjJJaWZR
Li5xQWd6aGdRaTVMMkdlaDl5V2RvN25nLmdMTHNpaTRjQk50c1NEUXRJTE51RWRnWTl4WkJVczJ4YzBaeTFGQjZQNmNzWWJf
ZkRnaDRZby04V1NaNjUzdXowbnFOalpXT1E1dGdnYW9qMlp6ZFk2ZzN2SWFHbF9JWUtNdkNIWWNscmt4YUFGTk5MWExLQlJm
aTA2LVk2V3l1dUdxNmpNWk5DbnlKX1pTbUpkVFQwc1Z4RTdGTXVxaUJsMElrRGdyVDdvOFNXMEY5cXFadndEZDJSaDdqNkRJ
WGdkS3VtOWltU2xNU1pjejhueVdic01Udk5yMWY0M25VenJzMHk5WWN6NnBDX0czZmlWYjJsX2VWLVFkcFh4TUo2bnZodXcy
djRiUGVkM3VMQlpaVW1oQ3B6TUVDdW5NMlh1TVBrTGdlS1NqWG44aGhPRFNVcW1WQ0Uta3RZdnRBc2Q0RnJxcGNxWlZiS0Zi
VTFRbU0wV2pMYVJtUk9IVllQVkc0a3FBdTRWalVMUzVCRWszNnZ4Nmp3U3BMUy1IdTcwbVRNcmR3dmV5Q2ZOYkhyT0FlVmVv
ekFIR3JqdGlmaFpmSFVUTWZiNkMtX2tOQVJGQWdDclZTZy0wUzlxb1JvTWVkUENETEE4MDJiaWwtNDJjOC15MWo4X1FVaC02
UUtCV2dodVd4VWtBODRpekFFaWl0QTlsSHFKM3Nxd2JFNURkZmhIay05bTJfTTN5MWlWVkdoRVQ3ZW9XVDBqWllnRGRBQjFz
UGwxLTlaSFNYYmsydTE3SkJVRV9FOXI0V0tWMnBqWGtiN0lQSWgtQ3JWQTZkcVdQRHVIbmx1V19wblNLYnYtTkZVbGQ0WEY3
cmZLYmQySlg4eUhhX05pOVVVUnUwZVdsNWxGRUVabklubmFKZEdHLUZrb3VuN2xHSFlwSE4ydXVudmRnOHZVZzZsa0JPbmoz
eUFjc1ZTMGxKc1NWdUxFYldwd2c4YjdBdDM3d3AtMWt2Y1ZQaWpCQ1lCV181d2JzbTFYd2k4MVc2WHVpNzMzQVg3cEJVQnBf
T2VRNzQ2ZXJJekNUUFZCYUpZUGJuZWEtdFhsU3RmZzBGeVRmbnhnX1Vzazl3QXJkemE4c204T0FQaWMxZmFQOG0uUTdFN0FV
X2xUVnNmZFI2bnkydUdhQSJ9.u2fJrVA55NQC3esPb4kcodt5rnjc1o-5uEDdUf-
KnCYEPBZ7t2CTsMMVVE3nfRhM39MfTlNS-qVOVpuoW_51NYaENXQMxfxlU9aXp944QiU1OeFQKj_g-
n2dEINRStbtUc3KMKqtz38BFf1g2Z51sdlnBn4XyVWPgGCf4XSfsFIa9fF051awQ0LcCv6YQTGer_6nk7t6F1MzPzBZzja1a
bpm--6LNSzjPftEiexpD2oXvW8Vl0Z9ggNk5Pn3Ne4RzqK09J9WChaJSXkTTE5G39EZcePmVNtcbayq-
L2pAK5weDa2k4uYMfAQAwcTOhUrwK3yilwqjHAamcG-CoiPZQ
```

```
OAuth Refresh Token Expiry Timer" parameter in enterprise parameters page in CUCM.
Path: System -> Enterprise parameters
Values are integers ranging from 1 - 90
Minimum lifetime = 1 Day
Default lifetime = 60 days
Maximum lifetime = 90 days
```

Jedes neue Zugriffstoken wird ausgegeben, wenn ein Client um ein Token bittet. Der alte behält seine Gültigkeit, solange:

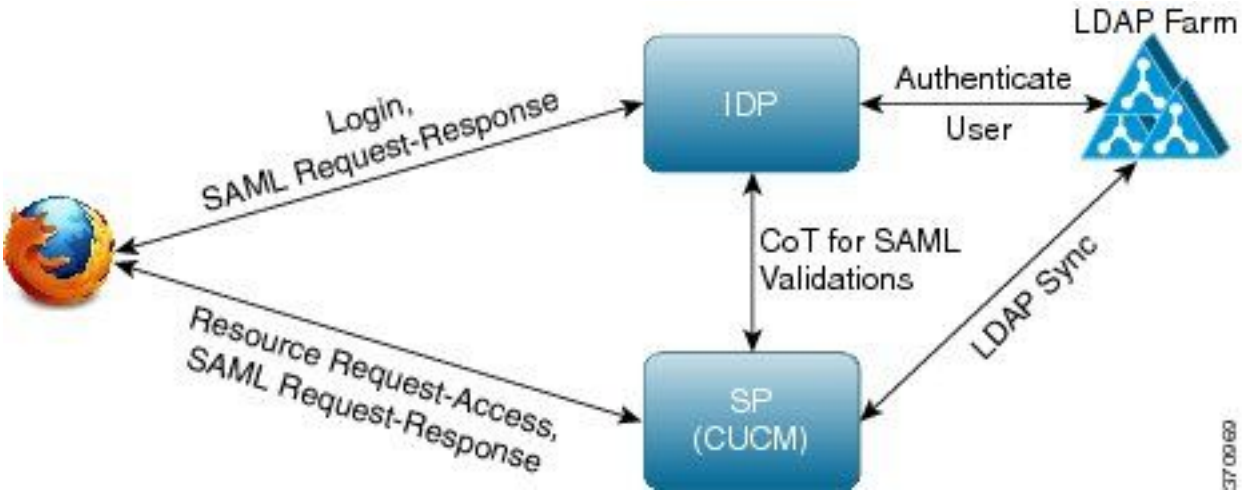
- Signatur-/Verschlüsselungsschlüssel wurden nicht geändert
- Die Gültigkeit (im Token gespeichert) bricht ab.
- JSON-Web-Token: bestehen aus drei durch Punkte getrennten Teilen, die: Header, Payload und Signatur.

Beispiel-Zugriffstoken:

- Am Anfang des fett hervorgehobenen Tokens befindet sich der Header.
- Mittlerer Teil ist die Payload.
- Wenn das Token am Ende fett markiert ist, dann ist es die Signatur.

Netzwerkdiagramm

Im Folgenden finden Sie eine grobe Übersicht über den Anrufablauf:



Aktualisierungstoken

- Aktualisierungstoken werden signiert.
- Aktualisierungstoken werden in der **aktualisiertokendetails-**Tabelle in der Datenbank als Hashwert selbst gespeichert. Dadurch soll die Replikation durch DB verhindert werden, da sie von jemandem ausgewählt werden kann. So überprüfen Sie die auszuführende Tabelle:
  `run sql select * from refreshtokendetails`
  oder mit einem lesbaren Gültigkeitsdatum:

```
run sql select pkid,refreshtokenindex,userid,clientid,dbinfo('utc_to_datetime',validity) as
validity,state from refreshtokendetails
```

```
admin:run sql select * from refreshtokendetails
pkid          refreshtokenindex userid   clientid validity            state
============= ================= ======== ======== ================== ==========
173e2283-1... 65483476618891... bvanturn C1b4b... 2019-01-05 14:11:46 1080686546
cd2c634c-7... 0bf6b2989db114... bvanturn C1b4b... 2019-01-05 14:28:41 569144456
a3706858-b... b4800f20dbfe0e... bvanturn C1b4b... 2019-01-05 14:38:12 1146722445
```

Warnung: Aktualisierungstoken werden von der DB geleert, wenn die Gültigkeit abgelaufen ist. Der Timer-Thread wird täglich um 2 Uhr ausgeführt (nicht über die Benutzeroberfläche konfigurierbar, kann aber über ein Remote-Support-Konto geändert werden). Wenn die Tabelle eine große Anzahl von Zugriffs-Token enthält, sind diese ungültig und müssen entfernt werden. Dies kann zu einer CPU-Spitze führen.

Sample refresh token:

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjhkMGQ1MzI0LWY0ZjAtNGIwYi04MTFlLTRhNTlmZGI2YjcyMjpj
Mjc3MGM5N2JkYTlkMzRmZDA1YTdlYTFhZWQzZTU0Y2E4MGJkZDdlZTM1ZDk3MDNiNjBiNTQ5MTBiZDQ0ODRiIn0.eyJleHAi
OjE1MDI2MjAwNTIsImlzcyI6IjhkMGQ1MzI0LWY0ZjAtNGIwYi04MTFlLTRhNTlmZGI2YjcyMiIsInR5cCI6InVzZXIiLCJ0
aWQiOiJiOTkxMjIxZi1mNDJlLTRlNTItODg3MS1jODc2ZTYzNWRkNWIiLCJjdHlwIjoicmVmcmVzaCIsImNjaWQiOiJDM2Iw
YWZmZWZlMTQzOTA0MTY4M2U5YzJjMzdkMzZmNDM4ZWYwZWYyN2MwOTM4YWRjNjIyNmUwYzAzZDE2OWYyYSJ9.creRusfwSYA
MAtttS2FIPAgIVvCiREvnzlouxeyGVndalJlMa-ZpRqv8FOBrsYwqEyulrl-
TeM8XGGQCUvFaqO9IkhJqSYz3zvFvvySWzDhl_pPyWIQteAhL1GaQkue6a5ZegeHRp1sjEczKMLC6H68CHCfletn5-
j2FNrAUOX99Vg5h4mHvlhfjJEel3dU_rciAIni12e3LOKajkzFxF6W0cXzzujyi2yPbY9gZsp9HoBbkkfThaZQbSlCEpvB3t
7yRfEMIEaHhEUU4M3-uSybuvitUWJnUIdTONiWGRh_fOFR9LV3Iv9J54dbsecpsncc369pYhu5IHwvsglNKEQ

Aktualisierungstoken aufrufen

Admin kann alle Aktualisierungstoken für einen Benutzer oder reine Geräte-Aktualisierungstoken für einen Benutzer über **Benutzer-ID** oder **Benutzer-ID** und **ClientID** widerrufen.

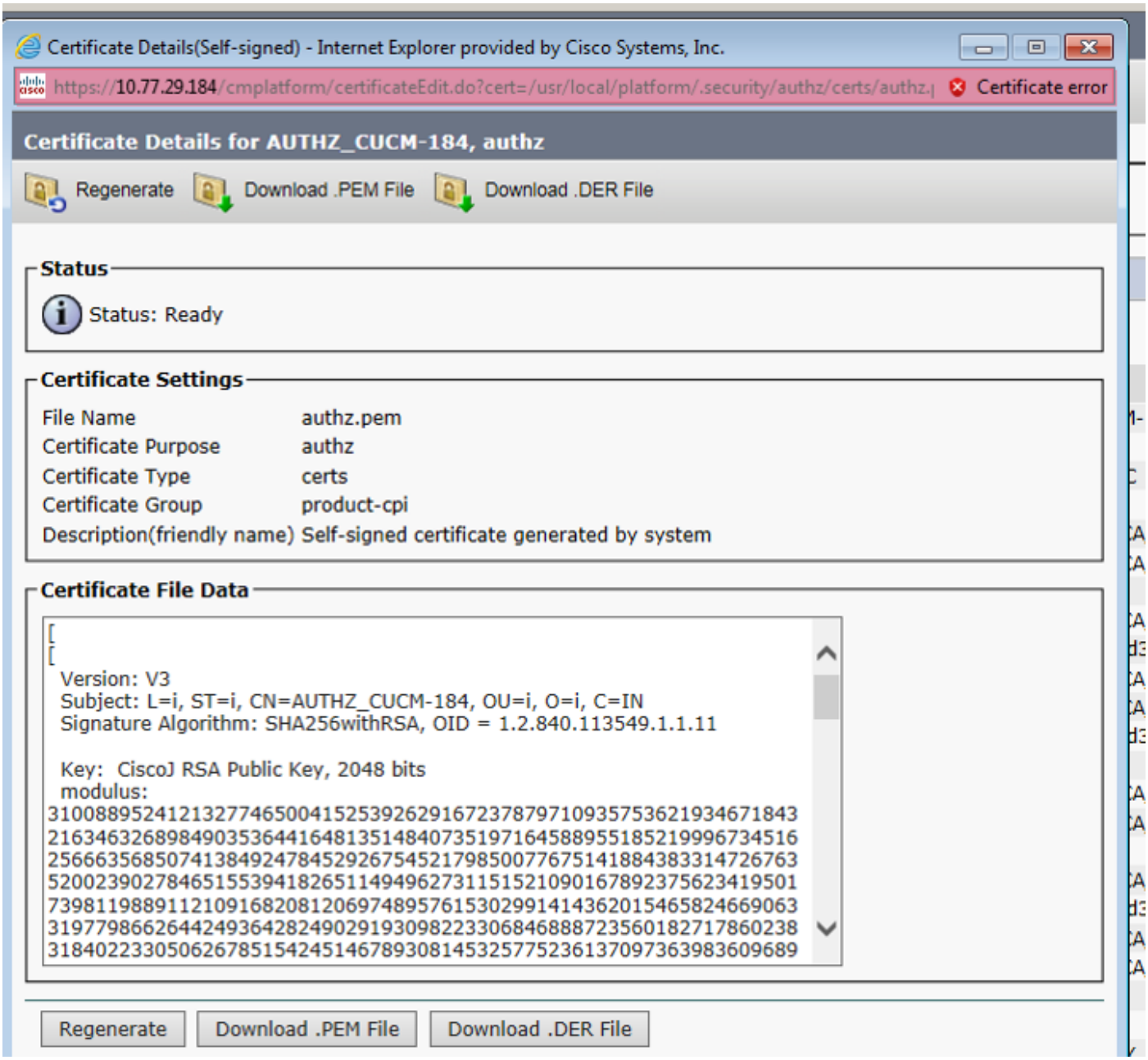
So widerrufen Sie gerätebasierte RTs für einen Benutzer:

- RT für Benutzer xyz und Gerät widerrufen, das durch client_id abc identifiziert wurde.
- https://cucm-193:8443/ssosp/token/revoke?user_id=xyz&client_id=abc

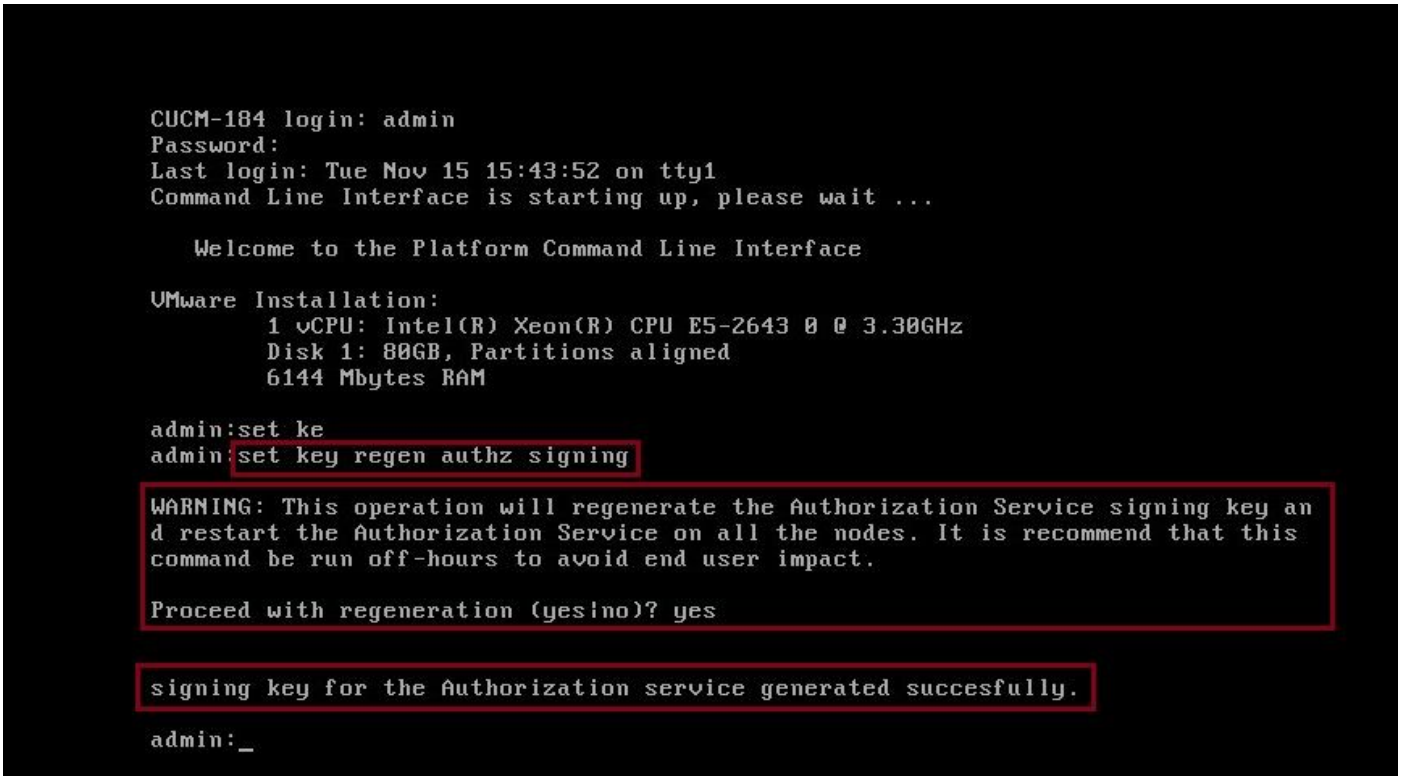
Signatur- und Verschlüsselungsschlüssel

- Der Signaturschlüssel basiert auf RSA und verfügt über ein Public/Private-Key-Paar.
- Der Verschlüsselungsschlüssel ist ein symmetrischer Schlüssel.
- Diese Schlüssel werden nur auf dem Publisher erstellt und auf alle Knoten im Cluster verteilt.
- Sowohl der Signaturschlüssel als auch der Verschlüsselungsschlüssel können mithilfe der aufgeführten Optionen neu generiert werden. Dies muss jedoch nur erfolgen, wenn der Administrator der Meinung ist, dass die Schlüssel kompromittiert wurden. Durch die erneute Generierung eines dieser Schlüssel werden alle vom AuthZ-Dienst ausgestellten Zugriffstoken ungültig.
- Signaturschlüssel können mit Benutzeroberfläche und CLI neu generiert werden.
- Verschlüsselungsschlüssel können nur mit CLI neu generiert werden.

Das Bild zeigt die Wiederherstellung von Autz-Zertifikaten (Signaturschlüssel) auf der Seite **Cisco Unified OS Administration** auf CUCM.



Die Wiederherstellung des Signaturschlüssels Authz mithilfe des CLI-Befehls wird im Bild gezeigt.



Admin kann mithilfe der CLI Authentifizierungs- und Verschlüsselungsschlüssel anzeigen. Der Hash des Schlüssels wird statt des ursprünglichen Schlüssels angezeigt.

Befehle zum Anzeigen von Schlüsseln sind:

Signaturschlüssel: **Schlüsselauthentifizierungssignierung anzeigen** und wie im Bild gezeigt.

```
admin:show key authz signing

authz signing key with checksum: a155d81be734850226f990a62816f1ae last synced on: 06/09/2017 13:04:47
```

Verschlüsselungsschlüssel: **Schlüsselauthentifizierungsverschlüsselung anzeigen** und wie im Bild gezeigt.

```
admin:show key authz encryption

authz encryption key with checksum: 88edce92173e33f9cedbbfb09cd0e8c4 last synced on: 06/14/2017 16:22:06
```

> **Anmerkung:** Die Authentisierung der Signierung und Verschlüsselung sind immer unterschiedlich.

Überprüfung

In diesem Abschnitt überprüfen Sie, ob Ihre Konfiguration ordnungsgemäß funktioniert.

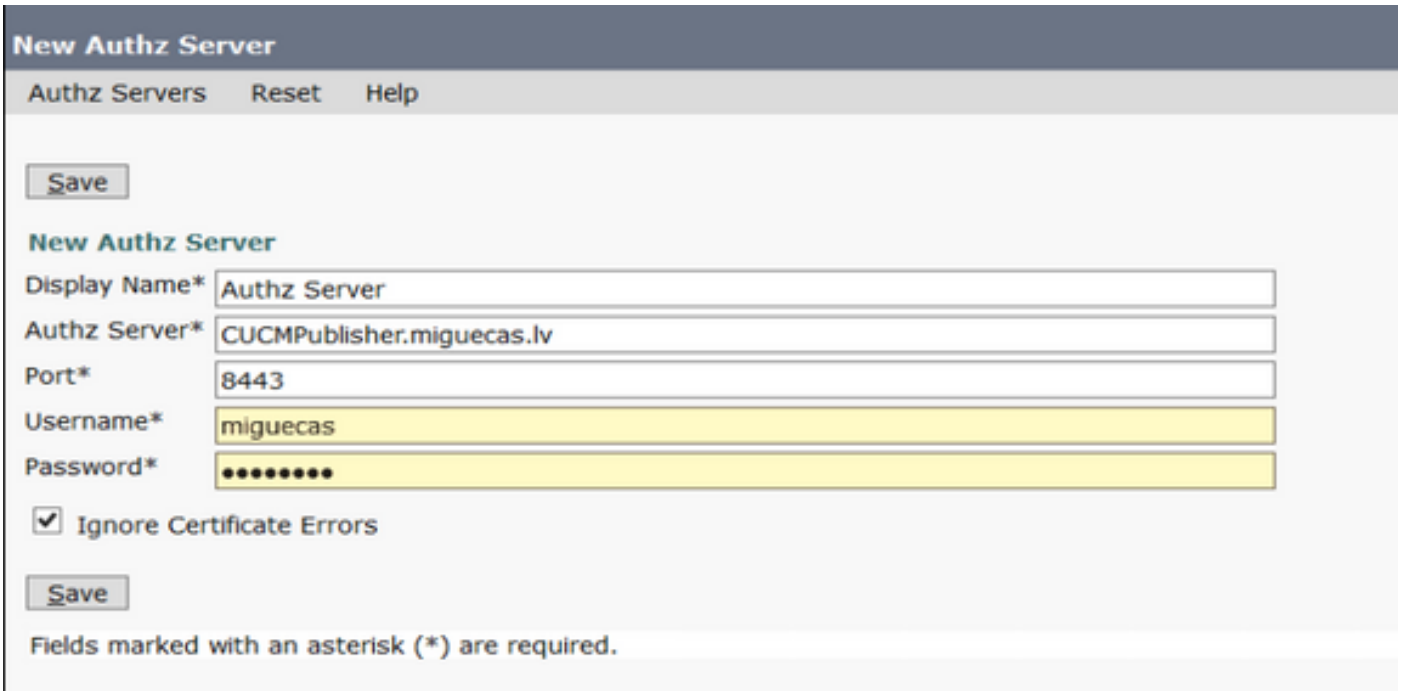
Wenn OAuth auf dem Cisco Unity Connection (CUC)-Server verwendet werden soll, muss der Netzwerkadministrator zwei Schritte ausführen.

Schritt 1: Konfigurieren Sie den Unity Connection Server so, dass die Token-Signierungs- und Verschlüsselungsschlüssel des OAuth vom CUCM abgerufen werden.

Schritt 2: Aktivieren Sie OAuth-Dienste auf dem CUC-Server.

> **Hinweis**: Um die Signierungs- und Verschlüsselungsschlüssel abzurufen, muss Unity mit den CUCM-Hostdetails konfiguriert werden, und ein Benutzerkonto, das für den CUCM AXL Access aktiviert ist. Wenn dies nicht konfiguriert ist, kann der Unity Server das OAuth-Token nicht vom CUCM abrufen, und die Voicemail-Anmeldung für die Benutzer kann nicht verfügbar sein.

Navigieren Sie zu **Cisco Unity Connection Administration > System Settings > Authz Servers.**



Fehlerbehebung

Dieser Abschnitt enthält Informationen, die Sie zur Fehlerbehebung bei Ihrer Konfiguration verwenden können.

> **Anmerkung:** Wenn OAuth verwendet wird und die Benutzer von Cisco Jabber sich nicht anmelden können, überprüfen Sie stets die Signierungs- und Verschlüsselungsschlüssel von den CUCM- und Instant Messaging and Presence (IM&P)-Servern.

Die Netzwerkadministratoren müssen diese beiden Befehle auf allen CUCM- und IM&P-Knoten **ausführen**:

- **Signierung der Schlüsselauthentifizierung anzeigen**
- **Verschlüsselung für die Schlüsselauthentifizierung anzeigen**

Wenn die Ausgaben für die Signaturauthentifizierung und die Verschlüsselungsauthentifizierung nicht mit allen Knoten übereinstimmen, müssen sie neu generiert werden. Dazu müssen diese beiden Befehle auf allen CUCM- und IM&P-Knoten ausgeführt werden:

- **Schlüsselauthentifizierungsverschlüsselung festlegen**
- **Signierung für Schlüsselauthentifizierung festlegen**

Anschließend muss der **Cisco Tomcat** Service auf allen Knoten neu gestartet werden.

Diese Fehlerzeile ist nicht nur in der Schlüsselübereinstimmung enthalten, sondern auch in den Cisco Jabber-Protokollen:

```
2021-03-30 14:21:49,631 WARN  [0x0000264c] [vices\impl\system\SingleSignOn.cpp(1186)] [Single-Sign-On-Logger] [CSFUnified::SingleSignOn::Impl::handleRefreshTokenFailure] - Failed to get valid access token from refresh token, maybe server issue.
```

Die so-App-Protokolle werden an folgenden Stellen generiert:

- **file view activelog platform/log/ssoApp.log** Dies erfordert keine Ablaufverfolgungskonfiguration für die Protokollauflistung. Bei jeder SSO-App-Operation werden in der Datei ssoApp.log neue Protokolleinträge generiert.
- SSOSP-Protokolle: **Dateiliste activelog tomcat/logs/ssosp/log4j**
  Bei jeder Aktivierung wird an diesem Speicherort eine neue Protokolldatei mit dem Namen **ssosp00XXX.log** erstellt. Alle anderen SSO-Vorgänge und alle Oauth-Vorgänge sind ebenfalls in dieser Datei angemeldet.
- Zertifikatprotokolle: **file list activelog platform/log/certMgmt*.log**
  Jedes Mal, wenn ein AuthZ-Zertifikat neu generiert wird (UI oder CLI), wird eine neue Protokolldatei für dieses Ereignis generiert.
  Für die erneute Generierung von Authentifizierungsschlüsseln wird für dieses Ereignis eine neue Protokolldatei generiert.

Zugehörige Informationen

Bereitstellung von OAuth mit Cisco Collaboration Solution, Version 12.0