Distribuzione E Risoluzione Dei Problemi Del Flusso Di Concessione Del Codice Di Autorizzazione - Miglioramento OAuth: Cisco Collaboration Solutions 12.0

Sommario

Introduzione

In questo documento viene descritto come il flusso di concessione del codice di autorizzazione si basa sul token di aggiornamento per migliorare l'esperienza utente di Jabber su vari dispositivi, in particolare per Jabber su Mobile.

Prerequisiti

Requisiti

Cisco raccomanda la conoscenza dei seguenti argomenti:

- Cisco Unified Communications Manager (CUCM) versione 12.0
- SSO (Single Sign On)/SAML
- Cisco Jabber
- Microsoft ADFS
- Provider di identità (IdP)

Per ulteriori informazioni su questi argomenti, fare riferimento ai seguenti collegamenti:

- Guida all'implementazione di SAML SSO per Cisco Unified Communications
- Esempio di configurazione di SAML SSO di Unified Communications Manager:
- Esempio di installazione di AD FS versione 2.0 per la configurazione di SAML SSO:

Componenti usati

Le informazioni fornite in questo documento si basano sul seguente software:

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

Le informazioni discusse in questo documento fanno riferimento a dispositivi usati in uno specifico ambiente di emulazione. Su tutti i dispositivi menzionati nel documento la configurazione è stata ripristinata ai valori predefiniti. Se la rete è operativa, valutare attentamente eventuali conseguenze derivanti dall'uso dei comandi.

Premesse

Attualmente, il flusso SSO di Jabber con Infrastructure si basa su un flusso di concessione implicito in cui il servizio CUCM Authz alloca i token di accesso di breve durata.

Dopo la scadenza del token di accesso, CUCM reindirizza Jabber a IdP per la riautenticazione.

Ciò comporta un'esperienza utente errata, in particolare con jabber su mobile, dove all'utente viene richiesto di immettere le credenziali frequentemente.

La soluzione per la riarchitettura della sicurezza propone inoltre il flusso di concessione del codice di autorizzazione (con l'utilizzo dell'approccio Refresh Tokens (estendibile agli endpoint/altre app di collaborazione)) per l'unificazione del flusso di accesso a Jabber e End Point per scenari SSO e non SSO.

Caratteristiche principali

- Il flusso di concessione del codice di autorizzazione è basato sul token di aggiornamento (estendibile a endpoint/altre app di collaborazione) per migliorare l'esperienza utente di Jabber in vari dispositivi, in particolare per Jabber su Mobile.
- Supporta token OAuth con firma e crittografia autonomi per consentire a diverse applicazioni di collaborazione di convalidare e rispondere alle richieste di risorse client.
- Il modello di flusso di concessione implicito viene mantenuto, consentendo la compatibilità con le versioni precedenti. Ciò consente anche un percorso senza interruzioni per altri client (come RTMT) che non sono passati al flusso di concessione del codice di autorizzazione.

Considerazioni importanti

- Implementazione tale che il vecchio client jabber possa funzionare con il nuovo CUCM (poiché supporta sia i flussi di concessione implicita che di concessione del codice di autorizzazione). Inoltre, il nuovo jabber può funzionare con il vecchio CUCM. Jabber può

determinare se CUCM supporta il flusso di concessione del codice di autorizzazione e solo se supporta questo modello, cambia e utilizza il flusso di concessione implicito.
- Il servizio AuthZ viene eseguito sul server CUCM.
- AuthZ supporta solo il flusso di concessione implicita. Ciò significa che non è presente alcun token di aggiornamento/token di accesso offline. Ogni volta che il client richiede un nuovo token di accesso, l'utente deve ripetere l'autenticazione con il provider di identità.
- I token di accesso sono stati emessi solo se la distribuzione è abilitata a SSO. Le distribuzioni non SSO non hanno funzionato in questo caso e i token di accesso non sono stati utilizzati in tutte le interfacce in modo coerente.
- I token di accesso non sono autonomi, ma vengono mantenuti nella memoria del server che li ha rilasciati. Se CUCM1 ha rilasciato il token di accesso, può essere verificato solo da CUCM1. Se il client tenta di accedere al servizio su CUCM2, CUCM2 deve convalidare tale token su CUCM1. Ritardi di rete (modalità proxy).
- L'esperienza utente sui client mobili è molto negativa, in quanto l'utente deve immettere nuovamente le credenziali su un tastierino alfanumerico quando esegue di nuovo l'autenticazione con l'IdP (in genere da 1 ora a 8 ore, che dipende da diversi fattori).
- I client che parlano a più applicazioni su più interfacce devono mantenere più credenziali/blocchi. Nessun supporto continuo per lo stesso accesso utente da 2 client simili. Ad esempio, l'utente A accede da istanze jabber che vengono eseguite su due diversi iPhone.
- AuthZ per supportare distribuzioni SSO e non SSO.
- AuthZ per supportare il flusso di concessione implicito + il flusso di concessione del codice di autorizzazione. Essendo **compatibile con le versioni precedenti,** consente a client come **RTMT** di continuare a lavorare fino a quando non si adattano.
- Con il flusso di concessione del codice di autorizzazione, AuthZ rilascia il token di accesso e il token di aggiornamento. Il token di aggiornamento può essere utilizzato per ottenere un altro token di accesso senza la necessità di autenticazione.
- I token di accesso sono autonomi, firmati e crittografati e utilizzano lo standard JWT (JSON web tokens) (conforme a RFC).
- Le chiavi di firma e crittografia sono comuni al cluster. Qualsiasi server nel cluster può verificare il token di accesso. Non è necessario mantenere la memoria.
- il servizio eseguito su CUCM 12.0 è il server di autenticazione centralizzato nel cluster.
- I token di aggiornamento sono memorizzati nel database (DB). L'amministratore deve essere in grado di revocarla, se necessario. La revoca si basa sull'ID utente o sull'ID utente e sull'ID client.
- I token di accesso firmati consentono a prodotti diversi di convalidare i token di accesso senza doverli archiviare. Durata configurabile del token di accesso e del token di aggiornamento (valore predefinito rispettivamente 1 ora e 60 giorni).
- Il formato JWT è allineato con Spark, che in futuro consente sinergie con i servizi Spark Hybrid.
- Supporto per lo stesso utente che accede da 2 dispositivi simili. Esempio: L'utente A può eseguire il login da istanze jabber che utilizzano due diversi iPhone.

Flusso di concessione elementi del codice di autorizzazione

- Auth Z Server
- Chiavi di crittografia
- Chiavi di firma

- Aggiorna token

Configurazione

Questa funzione non è attivata per impostazione predefinita.

Passaggio 1. Per abilitare questa funzione, selezionare **Sistema > Parametri aziendali**.

Passaggio 2. Impostare il parametro **OAuth con Aggiorna flusso di accesso** su Abilitato, come mostrato nell'immagine.

- Il token di accesso è firmato e crittografato. La chiave di firma e crittografia è comune al cluster. Ciò significa che qualsiasi nodo nel cluster può convalidare il token di accesso.
- Il token di accesso è in formato JWT (RFC 7519).
- I token di accesso riutilizzano il parametro enterprise (OAuth Access Token Expiry timer), applicabile sia ai vecchi formati di token che ai nuovi formati di token.
- Valore predefinito: 60 min.
- Valore minimo - 1 min.
- Valore Massimo - 1440 Minuti


```
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
```

Il nuovo token di accesso viene rilasciato ogni volta che il client ne richiede uno. La precedente rimane valida fino a quando:

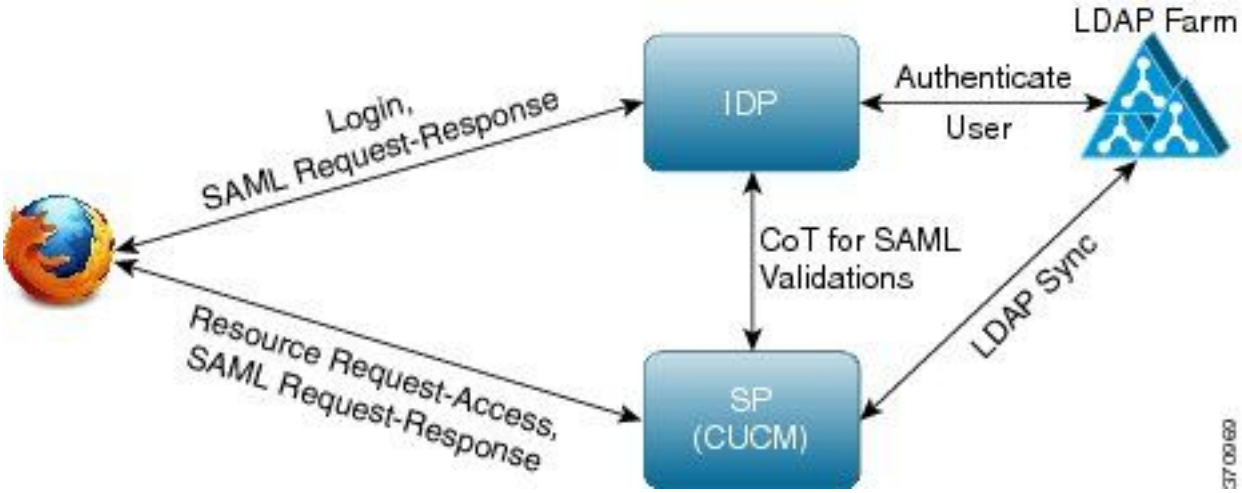
- Le chiavi di firma/crittografia non sono state modificate
- La validità (memorizzata nel token) viene interrotta.
- Token Web JSON: è costituita da tre parti, separate da punti, che sono: Intestazione, payload e firma.

Token di accesso di esempio:

- All'inizio del token evidenziato in grassetto è presente l'intestazione.
- La parte centrale è il Payload.
- Alla fine, se il token è evidenziato in grassetto, si tratta della Firma.

Esempio di rete

Di seguito è riportata una panoramica generale del flusso di chiamate interessato:



Aggiorna token

- Il token di aggiornamento è firmato.
- Il token di aggiornamento viene archiviato nella tabella **refreshtokendetails** nel database come valore hash di se stesso. In questo modo si impedisce la replica da parte del database, poiché può essere scelta da un altro utente. Per esaminare la tabella è possibile eseguire:
  ```
  run sql select * from refreshtokendetails
  ```
  Oppure con una data di validità leggibile:

  ```
  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
```

Avviso: Il token di aggiornamento viene scaricato dal database quando la validità è scaduta.

Il thread timer viene eseguito ogni giorno alle 2 del mattino (non è possibile configurarlo tramite l'interfaccia utente, ma è possibile modificarlo tramite l'account di supporto remoto). Se la tabella contiene un numero elevato di token di accesso, questi non sono validi e devono essere eliminati. Ciò può causare un picco della CPU.

Sample refresh token:


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


Revoca token di aggiornamento

L'amministratore ha la capacità di revocare tutti i token di aggiornamento per un utente o per un token di aggiornamento solo dispositivo per un utente tramite **userID** o **userID** e **ClientID**.

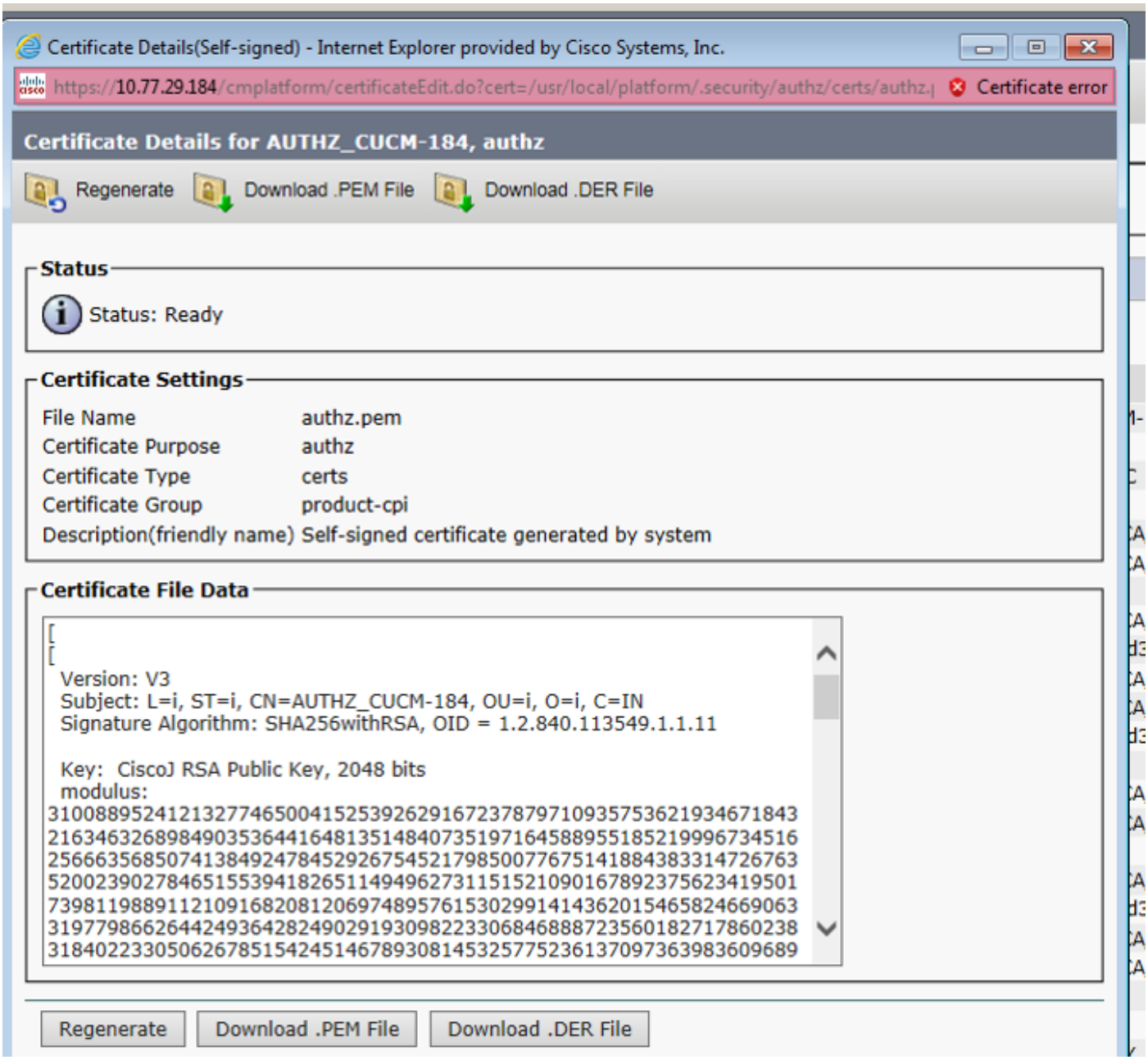
Per revocare gli RT basati su dispositivo per un utente:

- revocare la tecnologia RT per l'utente xyz e il dispositivo identificato da client_id abc.
- https://cucm-193:8443/ssosp/token/revoke?user_id=xyz&client_id=abc

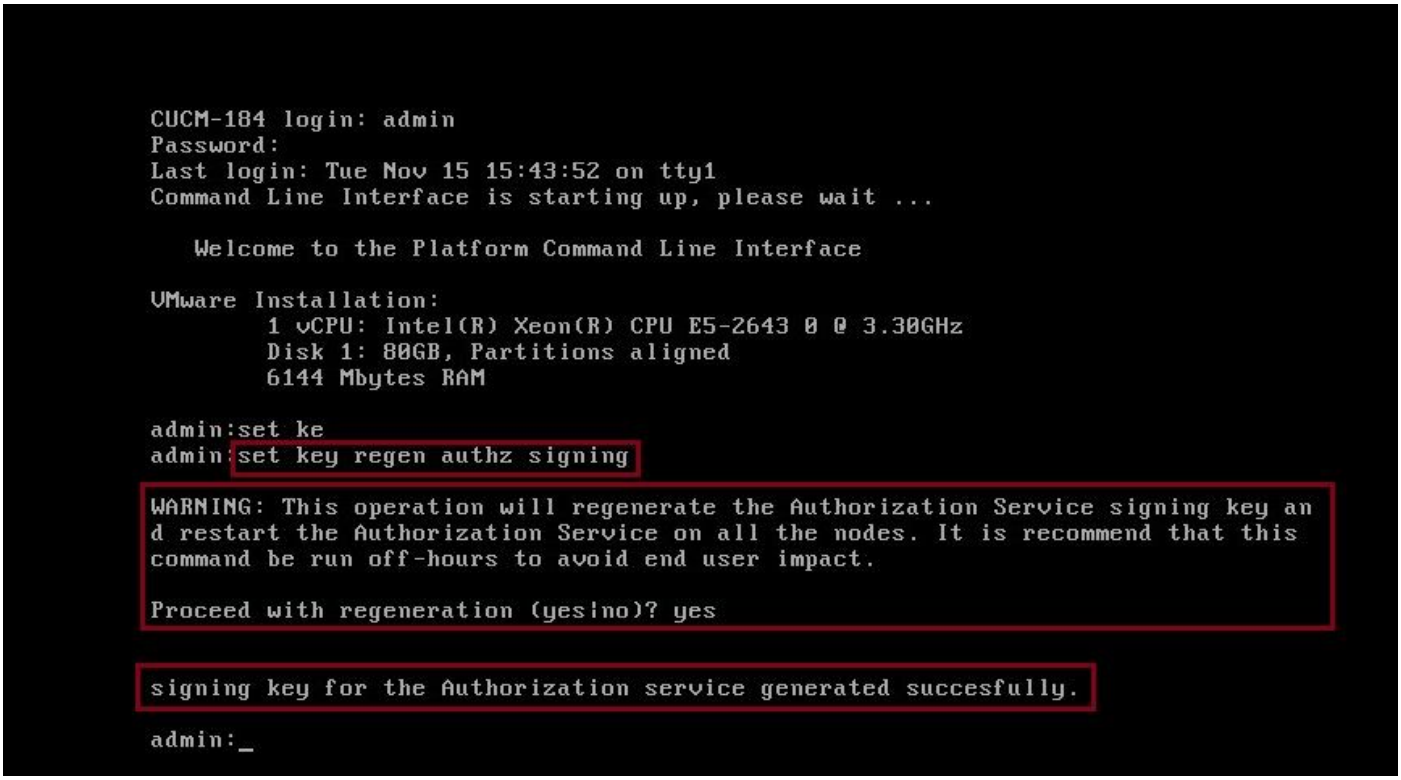
Chiavi di firma e crittografia

- La chiave di firma è basata su RSA, che dispone di una coppia di chiavi pubblica/privata.
- La chiave di crittografia è una chiave simmetrica.
- Queste chiavi vengono create solo nel server di pubblicazione e vengono distribuite in tutti i nodi del cluster.
- La chiave di firma e la chiave di crittografia possono essere rigenerate utilizzando le opzioni elencate. Tuttavia, questa operazione deve essere eseguita solo se l'amministratore ritiene che le chiavi siano state compromesse. L'impatto della rigenerazione di una di queste chiavi è che tutti i token di accesso rilasciati dal servizio AuthZ non sono più validi.
- Le chiavi di firma possono essere rigenerate con UI e CLI.
- Le chiavi di crittografia possono essere rigenerate solo con CLI.

La rigenerazione dei certificati Authz (chiave di firma) dalla pagina **Cisco Unified OS Administration** in CUCM è come mostrato nell'immagine.



La rigenerazione della chiave di firma Authz con l'uso del comando CLI è come mostrato nell'immagine.



L'amministratore può visualizzare le chiavi di firma e crittografia dell'autorizzazione tramite CLI. Viene visualizzato l'hash della chiave anziché la chiave originale.

Comando per la visualizzazione dei tasti:

Chiave firma: **mostra la firma dell'autorizzazione chiave** e come mostrato nell'immagine.

```
admin:show key authz signing

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

Chiave di crittografia: **show key authz encryption**, come mostrato nell'immagine.

```
admin:show key authz encryption

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

Nota: L'autorizzazione di firma e l'autorizzazione di crittografia sono sempre diverse.

Verifica

Fare riferimento a questa sezione per verificare che la configurazione funzioni correttamente.

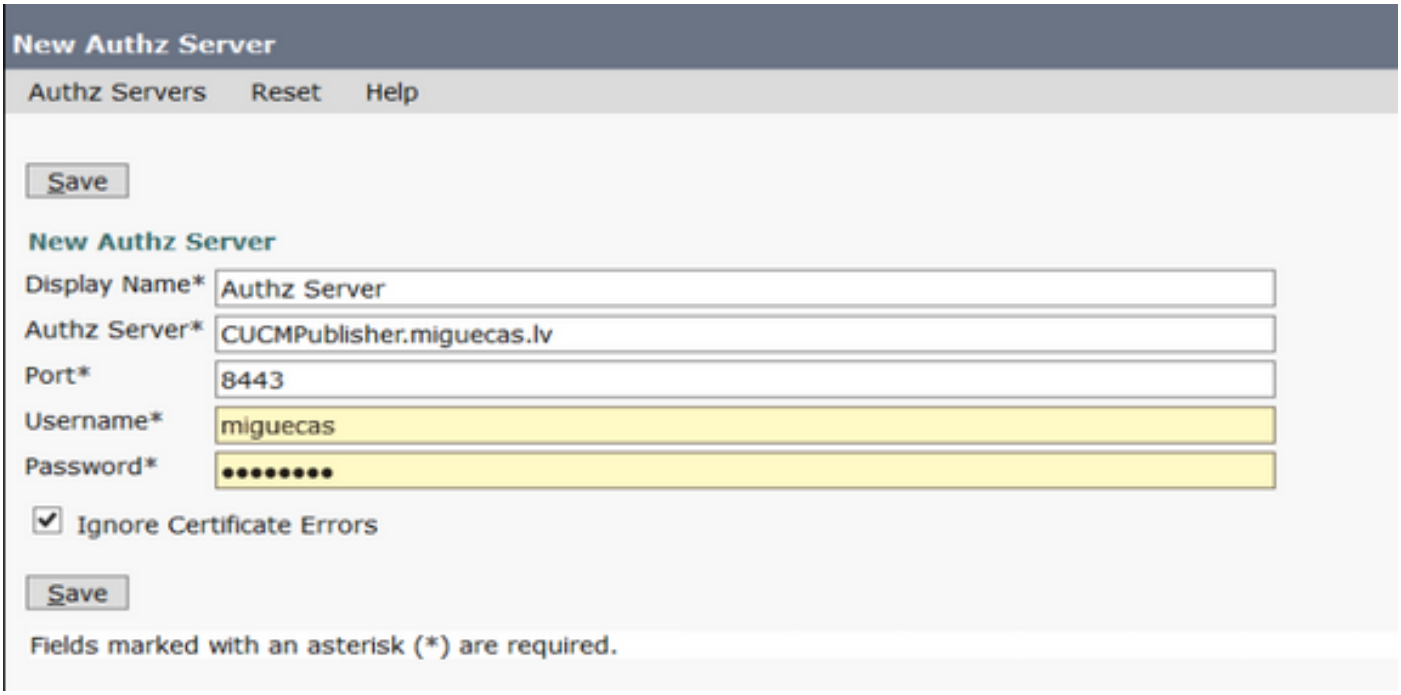
Quando si intende utilizzare OAuth sul server Cisco Unity Connection (CUC), l'amministratore di rete deve eseguire due passaggi.

Passaggio 1. Configurare Unity Connection Server in modo che recuperi le chiavi di crittografia e firma del token OAuth da CUCM.

Passaggio 2. Abilitare i servizi OAuth nel server CUC.

> **Nota**: per recuperare le chiavi di firma e crittografia, è necessario configurare Unity con i dettagli dell'host CUCM e un account utente abilitato per l'accesso AXL a CUCM. Se non è configurato, Unity Server non è in grado di recuperare il token OAuth da CUCM e il login della segreteria telefonica per gli utenti non può essere disponibile.

Selezionare **Cisco Unity Connection Administration > System Settings > Authz Servers** (Amministrazione connessione Cisco Unity > Impostazioni di sistema > Server di autenticazione)



Risoluzione dei problemi

Le informazioni contenute in questa sezione permettono di risolvere i problemi relativi alla configurazione.

> **Nota:** Se si utilizza OAuth e gli utenti di Cisco Jabber non sono in grado di accedere, controllare sempre le chiavi di firma e crittografia dei server CUCM e IM&P (Instant Messaging and Presence).

Gli amministratori di rete devono **eseguire** questi due comandi su tutti i nodi CUCM e IM&P:

- **mostra firma autenticazione chiave**
- **show key authz encryption**

Se gli output dell'autorizzazione di firma e dell'autorizzazione di crittografia non corrispondono in tutti i nodi, è necessario rigenerarli. Per eseguire questa operazione, è necessario eseguire questi due comandi su tutti i nodi CUCM e IM&P:

- **set key regen authz encryption**
- **imposta chiave regen authz**

In seguito, il servizio **Cisco Tomcat** deve essere riavviato su tutti i nodi.

Oltre alla mancata corrispondenza delle chiavi, nei log di Cisco Jabber è possibile trovare questa riga di errore:

```
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.
```

I log dell'app SSO vengono generati nei percorsi seguenti:

- **file view activelog platform/log/ssoApp.log** Non è necessaria alcuna configurazione di traccia per la raccolta dei log. Ogni volta che si esegue un'operazione di applicazione SSO, nel file ssoApp.log vengono generate nuove voci di log.
- Registri SSOSP: **elenco file activelog tomcat/logs/ssosp/log4j**
  Ogni volta che la funzione è abilitata, in questa posizione viene creato un nuovo file di log denominato **ssosp00XXX.log**. In questo file vengono inoltre registrate tutte le altre operazioni SSO e Oauth.
- Registri certificati: **elenco file activelog platform/log/certMgmt*.log**
  Ogni volta che viene rigenerato il certificato AuthZ (UI o CLI), viene generato un nuovo file di registro per questo evento.
  Per la rigenerazione delle chiavi di crittografia di autorizzazione, viene generato un nuovo file di registro per questo evento.

Informazioni correlate

Implementazione di OAuth con Cisco Collaboration Solution release 12.0