Dépannage de l'erreur Finesse « SSLPeerUncheckedException » pour les gadgets hébergés sur des serveurs signés CA

Contenu

Introduction

Ce document décrit les étapes pour dépanner le scénario où une chaîne de certificats signée par une autorité de certification (CA) est téléchargée vers Finesse pour un serveur Web externe qui héberge un gadget mais le gadget ne se charge pas quand vous vous connectez à Finesse et vous voyez l'erreur "SSLPeerUncheckedException".

Contribution de Gino Schweinsberger, ingénieur du centre d'assistance technique Cisco.

Conditions préalables

Conditions requises

Cisco vous recommande de prendre connaissance des rubriques suivantes :

- Certificats SSL
- administration Finesse
- Administration de Windows Server
- Analyse de capture de paquets avec Wireshark

Components Used

Les informations contenues dans ce document sont basées sur les versions de logiciel suivantes :

- Unified Contact Center Express (UCCX) 11.X
- Finesse 11.X

The information in this document was created from the devices in a specific lab environment. All of

the devices used in this document started with a cleared (default) configuration. Si votre réseau est en ligne, assurez-vous de bien comprendre l'incidence possible des commandes.

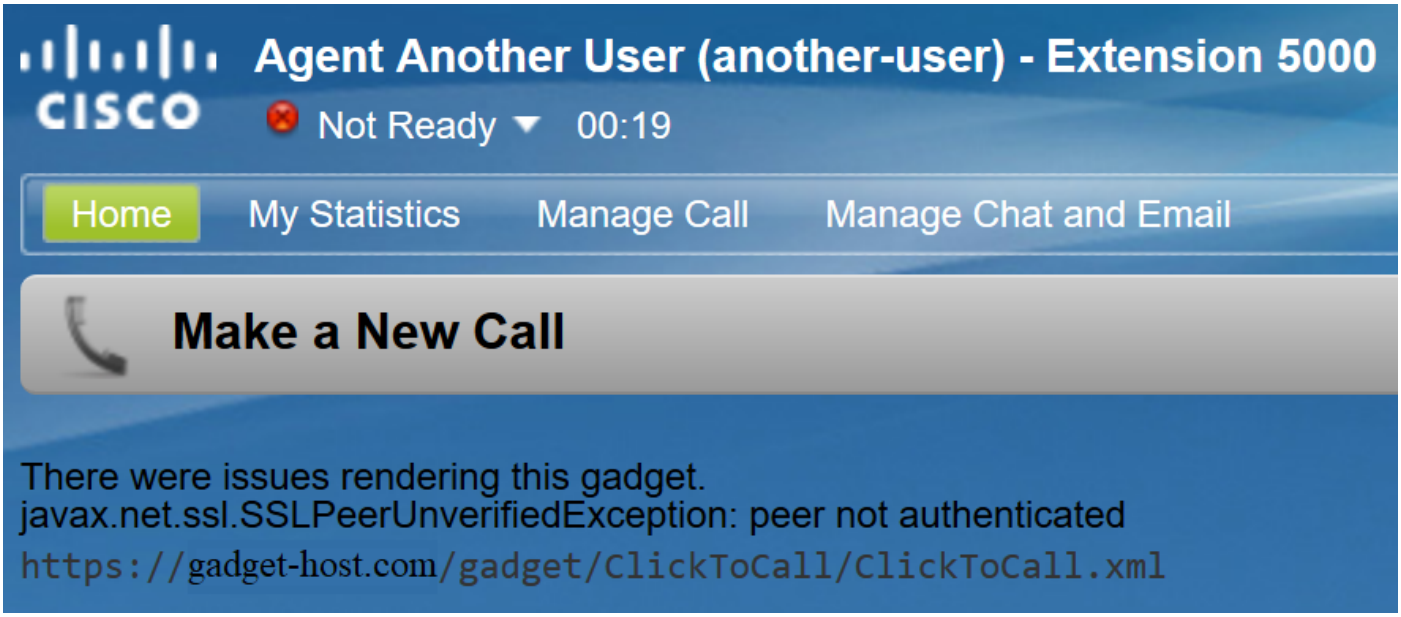
Informations générales

Voici les conditions pour que l'erreur se produise :

- Supposons que la chaîne de certificats de confiance est téléchargée dans Finesse
- Assurez-vous que les serveurs/services corrects ont été redémarrés
- Supposons que le gadget a été ajouté à la disposition Finesse avec une URL HTTPS et que l'URL est accessible

Voici l'erreur observée lorsque l'agent se connecte à Finesse :

«Des problèmes se sont produits lors du rendu de ce gadget.
javax.net.ssl.SSLPeerUncheckedException : homologue non authentifié"



Problèmes

Scénario 1 : Le serveur d'hébergement négocie les TLS non sécurisés

Lorsque Finesse Server effectue une demande de connexion au serveur d'hébergement, Finesse Tomcat annonce une liste de chiffrements de chiffrement qu'il prend en charge.

Certains chiffrements ne sont pas pris en charge en raison de failles de sécurité,

Si le serveur d'hébergement sélectionne l'un de ces chiffrements, la connexion est refusée :

- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA

Ces chiffrements sont connus pour utiliser des clés Diffie-Hellman éphémères faibles quand il négocie la connexion, et la vulnérabilité Logjam en fait un mauvais choix pour les connexions TLS.

Suivez le processus de connexion TLS dans une capture de paquets pour voir quel chiffre est négocié.

1. Finesse présente sa liste de chiffrements pris en charge dans l'étape **Client Hello** :

```
▾ TLSv1 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 67
  ▾ Handshake Protocol: Client Hello
      Handshake Type: Client Hello (1)
      Length: 63
      Version: TLS 1.0 (0x0301)
    › Random: 5cacb293b5efdb4cf1bb34464d7de9f5060b00a9beb81d29…
      Session ID Length: 0
      Cipher Suites Length: 24
    ▾ Cipher Suites (12 suites)
        Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
        Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)
        Cipher Suite: TLS_DHE_DSS_WITH_AES_256_CBC_SHA (0x0038)
        Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
        Cipher Suite: TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)
        Cipher Suite: TLS_DHE_DSS_WITH_AES_128_CBC_SHA (0x0032)
        Cipher Suite: TLS_RSA_WITH_RC4_128_SHA (0x0005)
        Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
        Cipher Suite: TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (0x0016)
        Cipher Suite: TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA (0x0013)
        Cipher Suite: TLS_RSA_WITH_RC4_128_MD5 (0x0004)
        Cipher Suite: TLS_EMPTY_RENEGOTIATION_INFO_SCSV (0x00ff)
      Compression Methods Length: 1
    › Compression Methods (1 method)
```

2. Pour cette connexion, **TLS_DHE_RSA_WITH_AES_256_CBC_SHA** a été sélectionné par le serveur d'hébergement au cours de l'étape **Hello du serveur,** car ce chiffre est plus élevé dans sa liste de chiffrements préférés.

```
TLSv1 Record Layer: Handshake Protocol: Multiple Handshake Messages
  Content Type: Handshake (22)
  Version: TLS 1.0 (0x0301)
  Length: 2557
  Handshake Protocol: Server Hello
    Handshake Type: Server Hello (2)
    Length: 77
    Version: TLS 1.0 (0x0301)
    Random: 5cacb292c4d7183627f620a066f9b6ce6460dcb849b59cae…
    Session ID Length: 32
    Session ID: 4c290000ce66098cc994a33e193b0da1244cb9f083f69c26…
    Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)
    Compression Method: null (0)
    Extensions Length: 5
    Extension: renegotiation_info (len=1)
  Handshake Protocol: Certificate
  Handshake Protocol: Server Key Exchange
    Handshake Type: Server Key Exchange (12)
    Length: 1032
    Diffie-Hellman Server Params
  Handshake Protocol: Server Hello Done
    Handshake Type: Server Hello Done (14)
    Length: 0
```

3. Finesse envoie une alerte Fatal et met fin à la connexion :

```
TLSv1 Record Layer: Alert (Level: Fatal, Description: Internal Error)
  Content Type: Alert (21)
  Version: TLS 1.0 (0x0301)
  Length: 2
  Alert Message
```

Solution

Afin d'empêcher l'utilisation de ces chiffrements, le serveur d'hébergement doit être configuré pour leur donner une faible priorité, ou ils doivent être complètement supprimés de la liste des chiffrements disponibles. Cela peut être fait sur un serveur Windows avec l'éditeur de stratégie de groupe Windows (gpedit.msc).

> **Note** : Pour plus de détails sur les effets de Logjam dans Finesse et l'utilisation de gpedit, vérifiez :

Scénario 2 : Le certificat a un algorithme de signature non pris en charge

Les autorités de certification Windows Server peuvent utiliser des normes de signature plus récentes pour signer des certificats. Bien qu'elle offre une sécurité supérieure à celle de SHA, l'adoption de ces normes en dehors des produits Microsoft est faible et les administrateurs risquent de rencontrer des problèmes d'interopérabilité.

Finesse Tomcat s'appuie sur le fournisseur de sécurité SunMSCAPI de Java pour activer la prise en charge des différents algorithmes de signature et fonctions cryptographiques utilisés par Microsoft. Toutes les versions actuelles de Java (1.7, 1.8 et 1.9) prennent uniquement en charge les algorithmes de signature suivants :

- MD5avecRSA
- MD2avec RSA
- AUCUNavecRSA
- SHA1 avec RSA
- SHA256avec RSA
- SHA384avec RSA
- SHA512avec RSA

Il est conseillé de vérifier la version de Java qui s'exécute sur le serveur Finesse pour vérifier quels algorithmes sont pris en charge dans cette version. La version peut être vérifiée à partir de l'accès racine avec cette commande : **java -version**

```
Using username "root".
Last login: Tue Apr 16 13:11:00 2019 from
[root@uccx12pub ~]# java -version
java version "1.7.0_181"
OpenJDK Runtime Environment (rhel-2.6.14.8.el6_9-i386 u181-b00)
OpenJDK Server VM (build 24.181-b00, mixed mode)
[root@uccx12pub ~]#
```

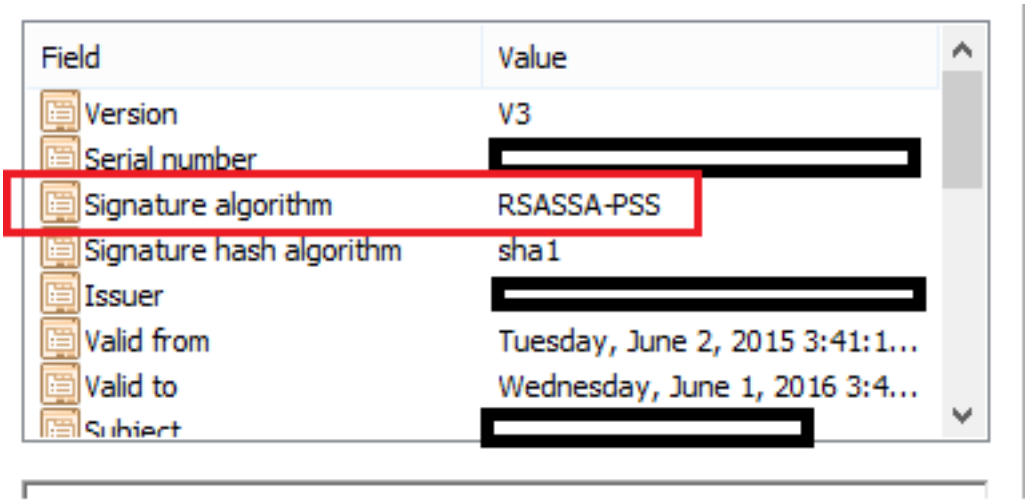
> **Remarque** : pour plus d'informations sur le fournisseur Java SunMSCAPI, consultez le site https://docs.oracle.com/javase/8/docs/technotes/guides/security/SunProviders.html#SunMSCAPI

Si un certificat est fourni avec une signature autre que celles répertoriées ci-dessus, Finesse ne peut pas utiliser le certificat pour créer une connexion TLS au serveur d'hébergement. Cela inclut les certificats qui sont signés avec un type de signature pris en charge, mais qui ont été émis par des autorités de certification qui ont leurs propres certificats intermédiaires et racine signés avec un autre élément.

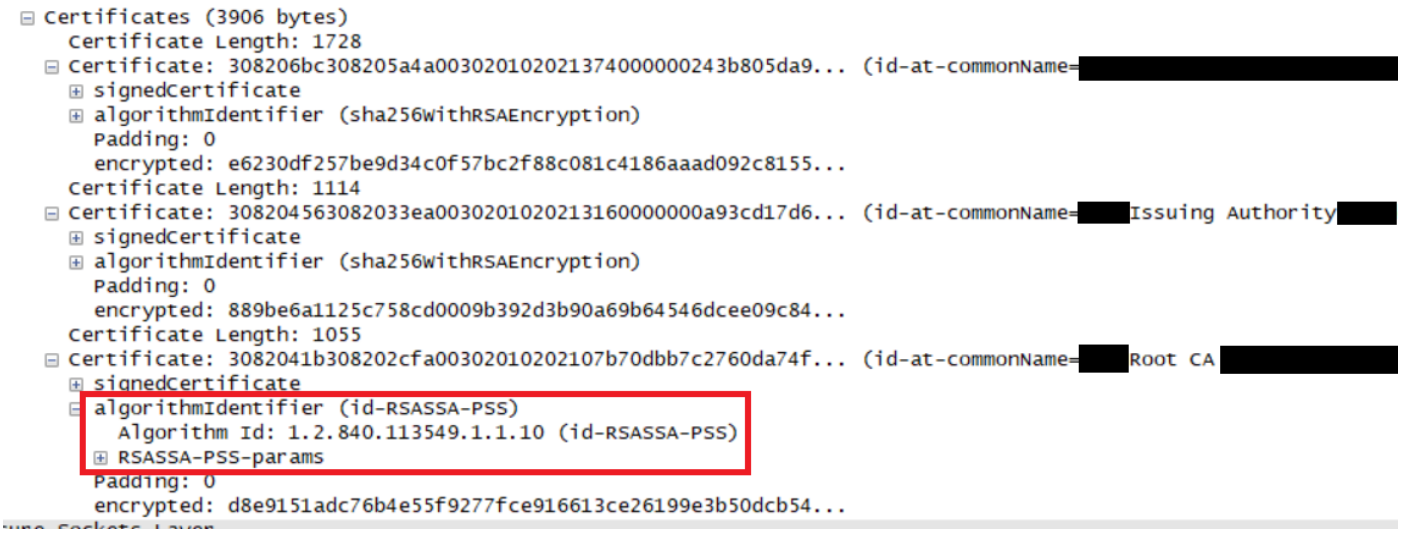
Si vous examinez une capture de paquets, Finesse ferme la connexion avec une « alerte fatale : Erreur « Certificate Unknown », comme illustré dans l'image.

```
Secure Sockets Layer
⊟ TLSv1.2 Record Layer: Alert (Level: Fatal, Description: Certificate Unknown)
    Content Type: Alert (21)
    Version: TLS 1.2 (0x0303)
    Length: 2
  ⊟ Alert Message
      Level: Fatal (2)
      Description: Certificate Unknown (46)
```

Àce stade, il est nécessaire de vérifier les certificats présentés par le serveur d'hébergement et de rechercher les algorithmes de signature non pris en charge. Il est courant de voir **RSASSA-PSS** comme l'algorithme de signature problématique :



Si un certificat de la chaîne est signé avec RSASSA-PSS, la connexion échoue. Dans ce cas, la capture de paquets montre que l'autorité de certification racine utilise RSASSA-PSS pour son propre certificat :



Solution

Pour résoudre ce problème, un nouveau certificat doit être émis par un fournisseur d'autorité de certification qui utilise uniquement l'un des types de signature SunMSCAPI pris en charge répertoriés dans toute la chaîne de certificats, comme expliqué précédemment.

> **Remarque** : pour plus d'informations sur l'algorithme de signature RSASSA-PSS, consultez le site https://pkisolutions.com/pkcs1v2-1rsassa-pss/

> **Note**: Ce problème est suivi dans le défaut CSCve79330

À propos de cette traduction
Cisco a traduit ce document en traduction automatisée vérifiée par une personne dans le cadre d'un service mondial permettant à nos utilisateurs d'obtenir le contenu d'assistance dans leur propre langue.

Il convient cependant de noter que même la meilleure traduction automatisée ne sera pas aussi précise que celle fournie par un traducteur professionnel.