Dépannez le " d'erreur de finesse ; SSLPeerUnverifiedException" ; pour des instruments hébergés sur les serveurs Ca-signés

Dépannez le " d'erreur de finesse ; SSLPeerUnverifiedException" ; pour des instruments hébergés sur les serveurs Ca-signés

Contenu

Introduction

Ce document décrit les étapes pour dépanner le scénario où un Autorité de certification (CA) - la chaîne de certificat signé est téléchargée à la finesse pour un web server externe qui héberge un instrument mais l'instrument ne charge pas quand vous ouvrez une session à la finesse et vous voyez l'erreur « SSLPeerUnverifiedException ».

Contribué par Gino Schweinsberger, ingénieur TAC Cisco.

Conditions préalables

Exigences

Cisco vous recommande de prendre connaissance des rubriques suivantes :

- Certificats SSL
- Gestion de finesse
- Gestion de Windows Server
- Analyse de capture de paquet avec Wireshark

Composants utilisés

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

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

Les informations contenues dans ce document ont été créées à partir des périphériques d'un environnement de laboratoire spécifique. Tous les périphériques utilisés dans ce document ont démarré avec une configuration effacée (par défaut). Si votre réseau est en ligne, assurez-vous de bien comprendre l'incidence possible des commandes.

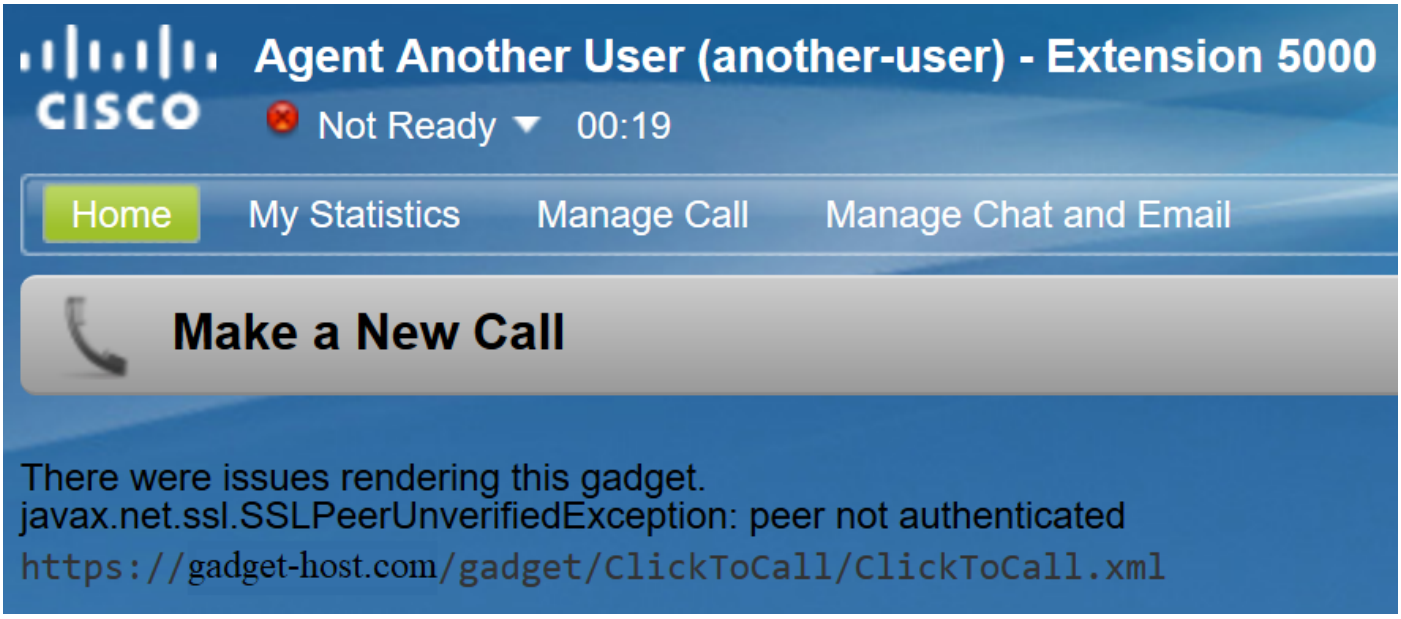
Informations générales

Ce sont les conditions pour que l'erreur se produise :

- Supposez que chaîne de confiance de certificat est téléchargé à la finesse
- Assurez-vous que les serveurs corrects/services ont été redémarrés
- Supposez que l'instrument a été ajouté à l'affichage de finesse avec un URL HTTPS et que l'URL est accessible

C'est l'erreur observée quand l'agent ouvre une session à la finesse :

« Il y avait des questions rendant cet instrument. javax.net.ssl.SSLPeerUnverifiedException : pair non authentifié »



Problèmes

Scénario 1 : Le serveur principal négocie le TLS unsecure

Quand le serveur de finesse fait une demande de connexion au serveur principal, la finesse Tomcat annonce une liste de chiffrements de cryptage qu'elle prend en charge.

Quelques chiffrements ne sont pas dus pris en charge aux failles de la sécurité,

Si le serveur principal sélectionne l'un ou l'autre 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 éphémères faibles de Diffie-Hellman quand il négocie la connexion, et la vulnérabilité d'embouteillage fait à ceux-ci un mauvais choix pour des connexions de TLS.

Suivez le processus de prise de contact de TLS dans une capture de paquet pour voir quel chiffrement est négocié.

1. La finesse présente sa liste de chiffrements pris en charge dans l'étape de **client bonjour** :

```
▾ 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 principal pendant l'étape de **serveur bonjour** parce que c'est plus élevé sur 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. La finesse envoie un vigilant mortel et finit 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 principal doit être configuré pour donner à ceux-ci une faible priorité, ou ils doivent être retirés de la liste de chiffrements disponibles complètement. Ceci peut être fait sur des Windows Server avec l'éditeur de stratégie de groupe de Windows (gpedit.msc).

> Remarque: Pour plus de détails sur les effets de l'embouteillage dans la finesse et l'utilisation du gpedit, contrôle :

Scénario 2 : Le certificat a un algorithme de signature sans support

Les autorités de certification de Windows Server peuvent employer de plus nouvelles normes de signature pour signer des Certificats. Même il offre la sécurité accrue que le SHA, l'adoption de ces normes en dehors de des Produits de Microsoft est basse et les administrateurs sont susceptibles de s'exécuter dans des problèmes d'interopérabilité.

La finesse Tomcat se fonde sur le fournisseur de Sécurité de SunMSCAPI de Javas pour activer le soutien des divers algorithmes de signature et des fonctions cryptographiques utilisés par Microsoft. Toutes les versions en cours de Javas (1.7, 1.8, et 1.9) prennent en charge seulement ces algorithmes de signature :

- MD5withRSA
- MD2withRSA
- NONEwithRSA
- SHA1withRSA
- SHA256withRSA
- SHA384withRSA
- SHA512withRSA

C'est une bonne idée de vérifier la version de Java qui fonctionne sur le serveur de finesse pour confirmer quels algorithmes sont pris en charge dans cette version. La version peut être vérifiée de l'accès de racine avec cette commande : **Javas - version**

```
Using username "root".
Last login: Tue Apr 16 13:11:00 2019 from
[root@uccxl2pub ~]# 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@uccxl2pub ~]#
```

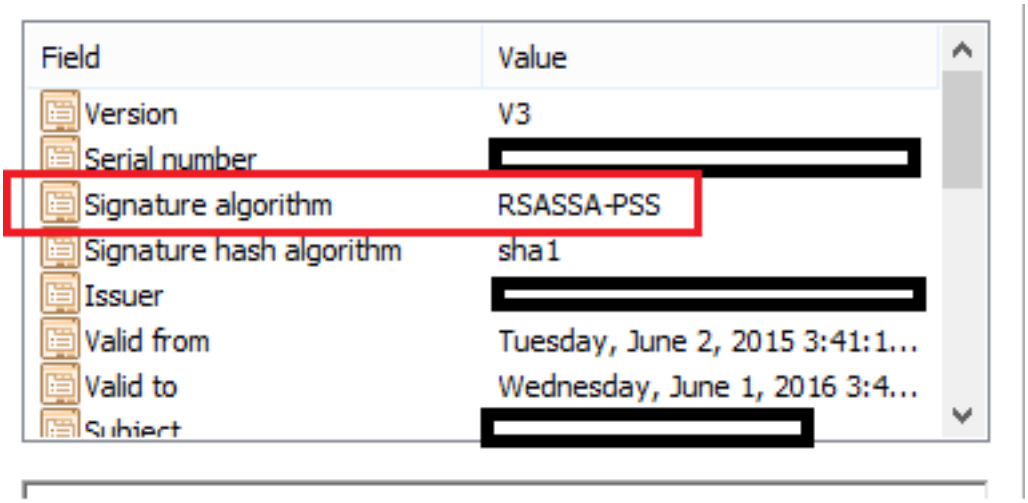
> Remarque: Pour plus de détails sur le fournisseur de SunMSCAPI de Javas référez-vous à https://docs.oracle.com/javase/8/docs/technotes/guides/security/SunProviders.html#SunMSCAPI

Si un certificat est équipé de signature autre que ceux répertoriés ci-dessus, la finesse ne peut pas employer le certificat pour créer une connexion de TLS au serveur principal. Ceci inclut les Certificats qui sont signés avec un type pris en charge de signature mais a été émis par les autorités de certification qui font signer leur propres intermédiaire et certificats racine avec autre chose.

Si vous regardez une capture de paquet, la finesse ferme la connexion avec « une alerte mortelle : Délivrez un certificat la » erreur inconnue, suivant les indications de 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)
```

En ce moment IS-IS nécessaire pour vérifier les Certificats présentés par le serveur principal et pour rechercher les algorithmes non vérifiés de signature. Il est commun pour voir **RSASSA-PSS** comme algorithme problématique de signature :



Si n'importe quel certificat dans la chaîne est signé avec RSASSA-PSS, la connexion échoue. Dans ce cas la capture de paquet prouve que la racine CA utilise RSASSA-PSS pour son propre certificat :

```
Certificates (3906 bytes)
    Certificate Length: 1728
  Certificate: 308206bc308205a4a00302010202137400000243b805da9... (id-at-commonName=[redacted]
    signedCertificate
    algorithmIdentifier (sha256WithRSAEncryption)
      Padding: 0
      encrypted: e6230df257be9d34c0f57bc2f88c081c4186aaad092c8155...
    Certificate Length: 1114
  Certificate: 308204563082033ea00302010202131600000000a93cd17d6... (id-at-commonName=[redacted]Issuing Authority[redacted]
    signedCertificate
    algorithmIdentifier (sha256WithRSAEncryption)
      Padding: 0
      encrypted: 889be6a1125c758cd0009b392d3b90a69b64546dcee09c84...
    Certificate Length: 1055
  Certificate: 3082041b308202cfa00302010202107b70dbb7c2760da74f... (id-at-commonName=[redacted]Root CA [redacted]
    signedCertificate
    algorithmIdentifier (id-RSASSA-PSS)
      Algorithm Id: 1.2.840.113549.1.1.10 (id-RSASSA-PSS)
    RSASSA-PSS-params
      Padding: 0
      encrypted: d8e9151adc76b4e55f9277fce916613ce26199e3b50dcb54...
```

Solution

Afin de résoudre ce problème, un nouveau certificat doit être délivré d'un fournisseur CA que seulement les utilisations une des types pris en charge de signature de SunMSCAPI ont répertorié dans tout la chaîne de certificat entière comme expliqué avant.

> Remarque: Pour plus de détails sur l'algorithme de signature RSASSA-PSS, voir le https://pkisolutions.com/pkcs1v2-1rsassa-pss/

> Remarque: Cette question est dépistée dans le défaut CSCve79330