Solucionar el error de Finesse "SSLPeerUnverifyException" para gadgets alojados en servidores firmados por CA

Contenido

Introducción

Este documento describe los pasos para resolver el escenario en el que una cadena de certificados firmada por la Autoridad de Certificación (CA) se carga en Finesse para un servidor web externo que aloja un gadget, pero el gadget no se carga cuando se inicia sesión en Finesse y aparece el error "SSLPeerUnverifyException".

Colaboración de Gino Schweinsberger, ingeniero del TAC de Cisco.

Prerequisites

Requirements

Cisco recomienda que tenga conocimiento sobre estos temas:

- Certificados SSL
- Administración Finesse
- administración de Windows Server
- Análisis de captura de paquetes con Wireshark

Componentes Utilizados

La información que contiene este documento se basa en estas versiones de software:

- 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 tiene una red en vivo, asegúrese de entender el posible impacto de cualquier comando.

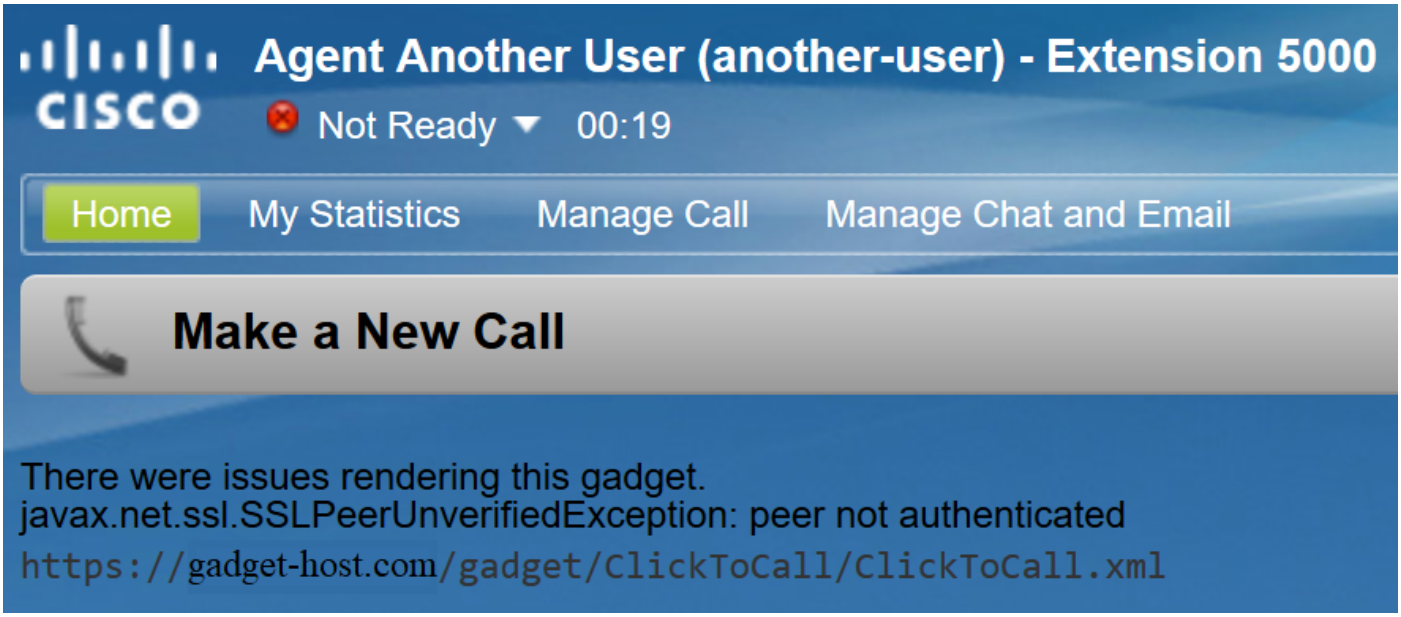
Antecedentes

Estas son las condiciones para que se produzca el error:

- Suponer que la cadena de confianza de certificados se carga en Finesse
- Asegúrese de que se han reiniciado los servidores/servicios correctos
- Suponga que el gadget se ha agregado al diseño Finesse con una URL HTTPS y que la URL es accesible

Este es el error observado cuando el agente inicia sesión en Finesse:

"Hubo problemas al procesar este gadget. javax.net.ssl.SSLPeerUnverifyException: peer no autenticado"



Problemas

Escenario 1: El servidor de alojamiento negocia TLS no seguro

Cuando Finesse Server realiza una solicitud de conexión al servidor de alojamiento, Finesse Tomcat anuncia una lista de cifrados de cifrado que admite.

Algunos cifrados no son compatibles debido a vulnerabilidades de seguridad,

Si el servidor de alojamiento selecciona cualquiera de estos cifrados, se rechaza la conexión:

- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA

Se sabe que estos cifrados utilizan claves Diffie-Hellman efímeras débiles cuando negocian la conexión, y la vulnerabilidad Logjam los convierte en una mala opción para las conexiones TLS.

Siga el proceso de intercambio de señales TLS en una captura de paquetes para ver qué cifrado se negocia.

1. Finesse presenta su lista de cifrados soportados en el paso **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. Para esta conexión, **TLS_DHE_RSA_WITH_AES_256_CBC_SHA** fue seleccionado por el servidor de alojamiento durante el paso **Server Hello** porque es superior en su lista de cifrados preferidos.

```
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 envía una alerta Fatal y finaliza la conexión:

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

Solución

Para evitar el uso de estos cifrados, el servidor de alojamiento debe configurarse para darles una prioridad baja o deben eliminarse completamente de la lista de cifrados disponibles. Esto se puede hacer en un servidor Windows con el Editor de directivas de grupo de Windows (gpedit.msc).

> **Nota**: Para más detalles sobre los efectos de Logjam en Finesse y el uso de gpedit, consulte:

Escenario 2: El certificado tiene un algoritmo de firma no admitido

Las autoridades de certificados de Windows Server pueden utilizar estándares de firma más recientes para firmar certificados. Incluso si ofrece mayor seguridad que SHA, la adopción de estos estándares fuera de los productos de Microsoft es baja y es probable que los administradores tengan problemas de interoperabilidad.

Finesse Tomcat confía en el proveedor de seguridad SunMSCAPI de Java para habilitar la compatibilidad con los diversos algoritmos de firma y funciones criptográficas que utiliza Microsoft. Todas las versiones actuales de Java (1.7, 1.8 y 1.9) sólo admiten estos algoritmos de firma:

- MD5conRSA
- MD2conRSA
- NONEwithRSA
- SHA1conRSA
- SHA256conRSA
- SHA384conRSA
- SHA512conRSA

Es una buena idea verificar la versión de Java que se ejecuta en el servidor Finesse para confirmar qué algoritmos se soportan en esa versión. La versión se puede verificar desde el acceso raíz con este comando: **java -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 ~]#
```

> **Nota**: para obtener más información sobre el proveedor SunMSCAPI de Java, consulte https://docs.oracle.com/javase/8/docs/technotes/guides/security/SunProviders.html#SunMSCAPI

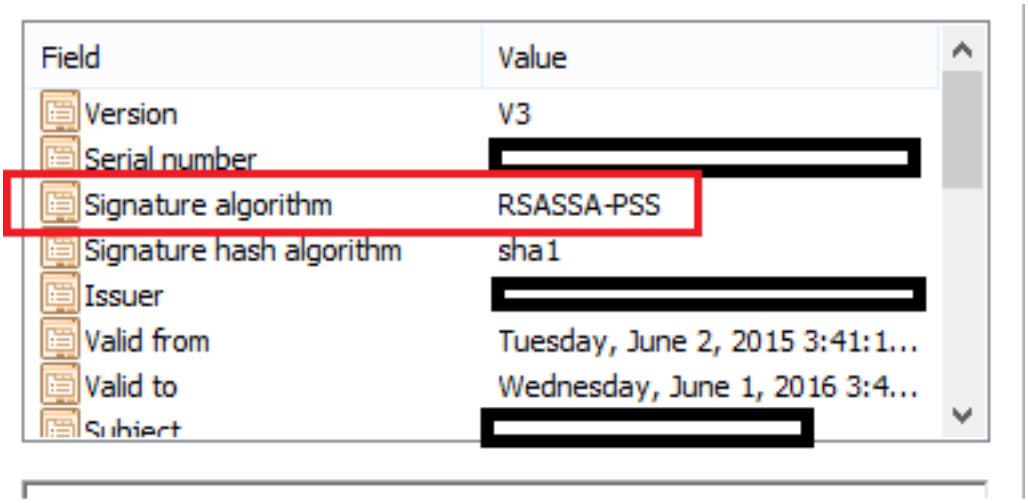
Si se proporciona un certificado con una firma distinta de las enumeradas anteriormente, Finesse no podrá utilizar el certificado para crear una conexión TLS con el servidor de alojamiento. Esto incluye los certificados firmados con un tipo de firma compatible pero emitidos por autoridades de certificados que tienen sus propios certificados raíz e intermedios firmados con otra cosa.

Si observa una captura de paquetes, Finesse cierra la conexión con una "alerta fatal: Error "Certificado desconocido", como se muestra en la imagen.

```
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 este punto es necesario comprobar los certificados presentados por el servidor de alojamiento

y buscar algoritmos de firma no admitidos. Es común ver **RSASA-PSS** como el algoritmo de firma problemática:



Si algún certificado de la cadena está firmado con RSASA-PSS, la conexión falla. En este caso, la captura de paquetes muestra que la CA raíz utiliza RSASA-PSS para su propio certificado:

Solución

Para resolver este problema, se debe emitir un nuevo certificado desde un proveedor de CA que solo utilice uno de los tipos de firma SunMSCAPI admitidos que se muestran en toda la cadena de certificados, como se explicó anteriormente.

> **Nota**: para obtener más información sobre el algoritmo de firma RSASA-PSS, consulte https://pkisolutions.com/pkcs1v2-1rsassa-pss/

> **Nota:** Este problema se rastrea en el defecto CSCve79330

Acerca de esta traducción
Cisco ha traducido este documento combinando la traducción automática y los recursos humanos a fin de ofrecer a nuestros usuarios en todo el mundo contenido en su propio idioma.

Tenga en cuenta que incluso la mejor traducción automática podría no ser tan precisa como la proporcionada por un traductor profesional.

Cisco Systems, Inc. no asume ninguna responsabilidad por la precisión de estas traducciones y recomienda remitirse siempre al documento original escrito en inglés (insertar vínculo URL).