Risoluzione dei problemi relativi all'errore Finesse "SSLPeerUnverifyException" per i gadget ospitati sui server con firma CA

Sommario

[Introduzione](#)
[Prerequisiti](#)
[Requisiti](#)
[Componenti usati](#)
[Premesse](#)
[Problemi](#)
[Scenario 1: Il server di hosting negozia un TLS non sicuro](#)
[Soluzione](#)
[Scenario 2: Il certificato contiene un algoritmo di firma non supportato](#)
[Soluzione](#)

Introduzione

In questo documento vengono descritti i passaggi per la risoluzione dei problemi relativi allo scenario in cui una catena di certificati firmata da CA (Certification Authority) viene caricata in Finesse per un server Web esterno che ospita un gadget ma il gadget non viene caricato quando si accede a Finesse e viene visualizzato l'errore "SSLPeerUnverifyException".

Contributo di Gino Schweinsberger, Cisco TAC Engineer.

Prerequisiti

Requisiti

Cisco raccomanda la conoscenza dei seguenti argomenti:

- Certificati SSL
- Amministrazione Finesse
- Amministrazione di Windows Server
- Analisi della cattura dei pacchetti con Wireshark

Componenti usati

Le informazioni di questo documento si basano sulle seguenti versioni software:

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

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.

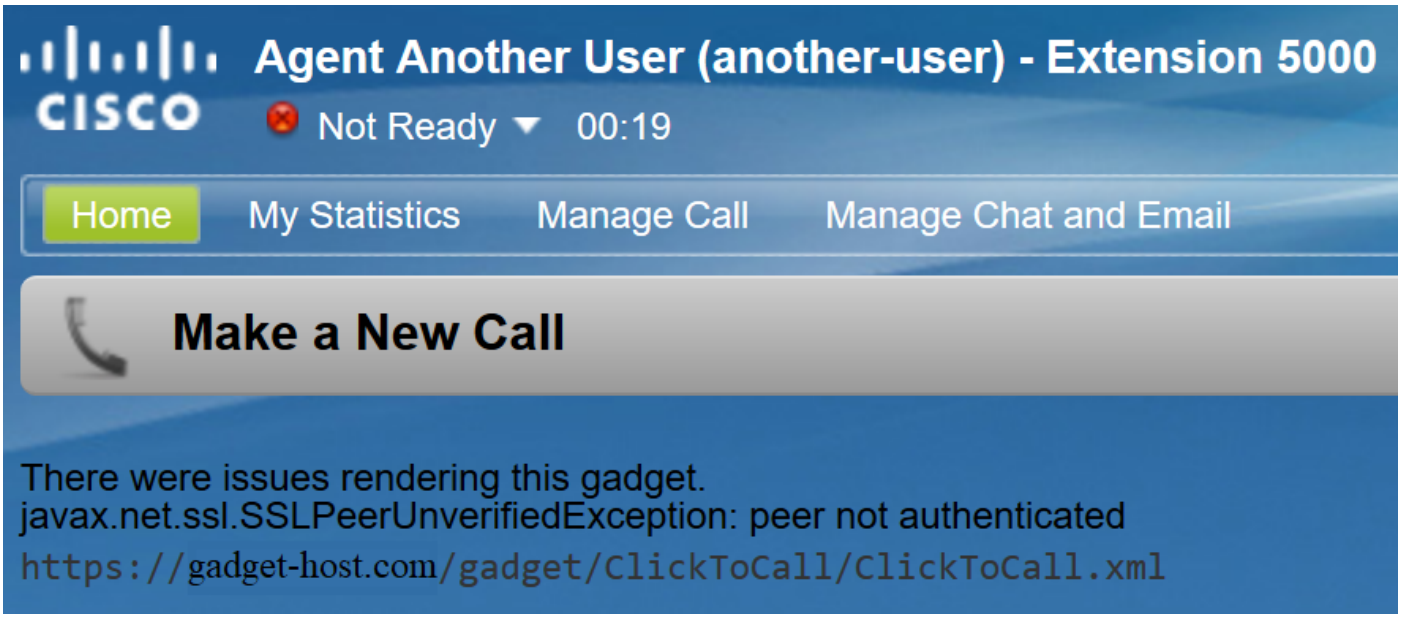
Premesse

Le condizioni dell'errore sono le seguenti:

- Si supponga che la catena di certificati sia caricata in Finesse
- Verificare che i server/servizi corretti siano stati riavviati
- Si supponga che il gadget sia stato aggiunto al layout Finesse con un URL HTTPS e che l'URL sia raggiungibile

Questo è l'errore osservato quando l'agente accede a Finesse:

"Si sono verificati problemi durante il rendering di questo gadget.
javax.net.ssl.SSLPeerUnverifyException: peer non autenticato"



Problemi

Scenario 1: Il server di hosting negozia un TLS non sicuro

Quando Finesse Server effettua una richiesta di connessione al server di hosting, Finesse Tomcat annuncia un elenco di cifrari di crittografia che supporta.

Alcune cifrature non sono supportate a causa di vulnerabilità della sicurezza,

Se il server di hosting seleziona una di queste cifrature, la connessione viene rifiutata:

- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA

Ènoto che questi cifrari utilizzano chiavi Diffie-Hellman effimere deboli quando negoziano la connessione, e la vulnerabilità Logjam li rende una cattiva scelta per le connessioni TLS.

Seguire il processo di handshake TLS in un'acquisizione pacchetto per verificare quale cifratura viene negoziata.

1. Finesse presenta l'elenco delle cifrature supportate nel passaggio **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. Per questa connessione, **TLS_DHE_RSA_WITH_AES_256_CBC_SHA** è stato selezionato dal server di hosting durante il passaggio **Server Hello** perché è più in alto nella lista delle cifrature preferite.

```
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 invia un avviso di errore irreversibile e termina la connessione:

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

Soluzione

Per impedire l'utilizzo di questi cifrari, il server di hosting deve essere configurato in modo da assegnare loro una priorità bassa oppure devono essere rimossi completamente dall'elenco di cifrari disponibili. Questa operazione può essere eseguita in un server Windows con l'editor di Criteri di gruppo di Windows (gpedit.msc).

> **Nota**: per ulteriori dettagli sugli effetti di Logjam in Finesse e l'uso di gpedit, controllare:

Scenario 2: Il certificato contiene un algoritmo di firma non supportato

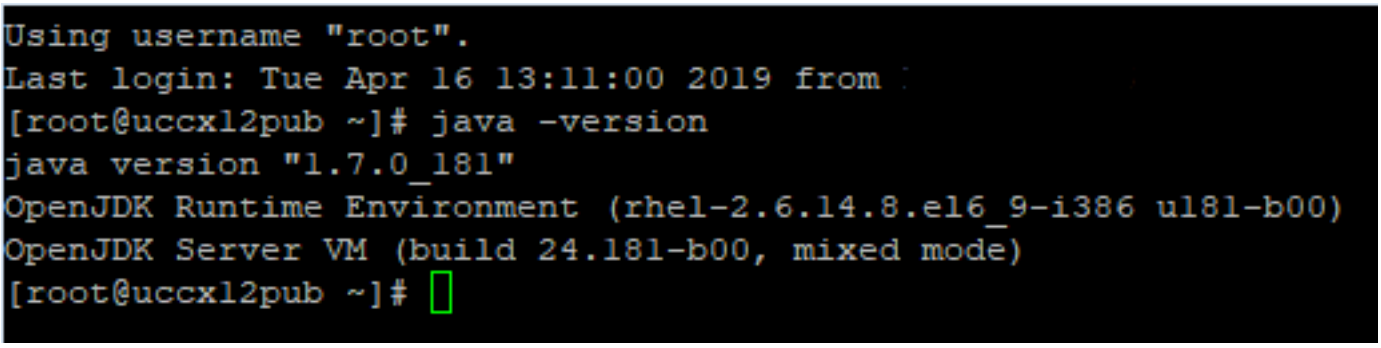
Le autorità di certificazione di Windows Server possono utilizzare standard di firma più recenti per

firmare i certificati. Anche se offre una maggiore sicurezza rispetto a SHA, l'adozione di questi standard al di fuori dei prodotti Microsoft è bassa e gli amministratori probabilmente incontreranno problemi di interoperabilità.

Finesse Tomcat si affida al provider di protezione SunMSCAPI di Java per abilitare il supporto per i vari algoritmi di firma e le funzioni di crittografia utilizzati da Microsoft. Tutte le versioni correnti di Java (1.7, 1.8 e 1.9) supportano solo questi algoritmi di firma:

- MD5 con RSA
- MD2 con RSA
- NONEwithRSA
- SHA1 con RSA
- SHA256conRSA
- SHA384 con RSA
- SHA512con RSA

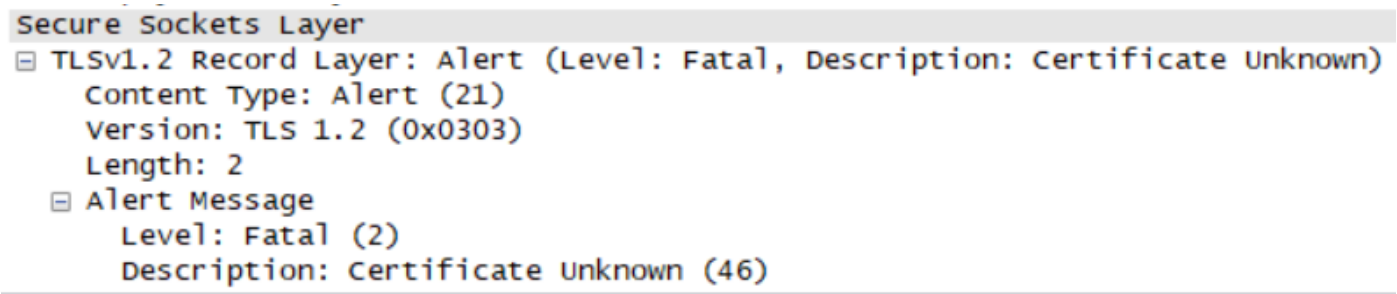
Èconsigliabile controllare la versione di Java in esecuzione sul server Finesse per verificare quali algoritmi sono supportati in tale versione. Per controllare la versione dall'accesso alla directory principale, usare questo comando: **java - versione**



> **Nota**: per ulteriori informazioni sul provider Java SunMSCAPI, visitare il sito Web all'indirizzo https://docs.oracle.com/javase/8/docs/technotes/guides/security/SunProviders.html#SunMSCAPI

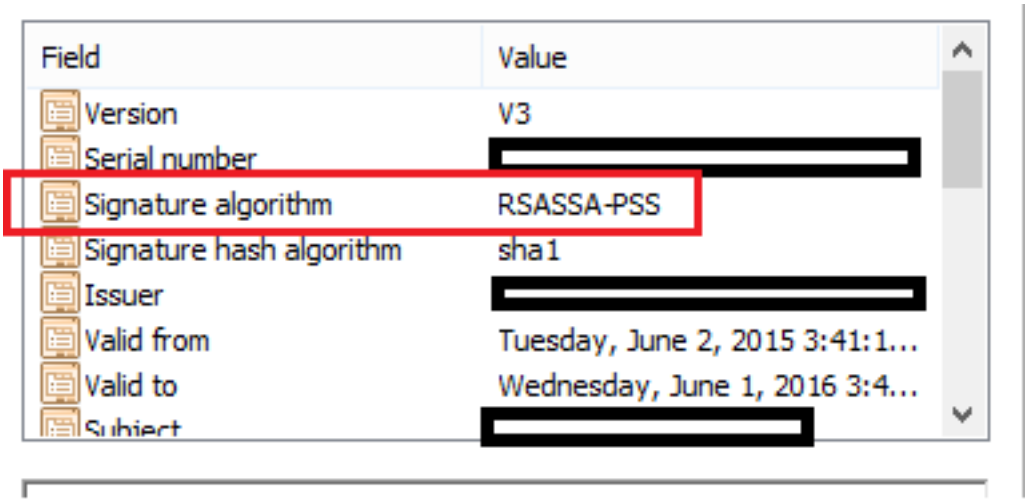
Se un certificato viene fornito con una firma diversa da quelle elencate sopra, Finesse non è in grado di utilizzare il certificato per creare una connessione TLS al server di hosting. Sono inclusi i certificati firmati con un tipo di firma supportato, ma emessi da autorità di certificazione che dispongono di propri certificati intermedi e radice firmati con qualcos'altro.

Se si osserva un'acquisizione di pacchetti, Finesse chiude la connessione con un "avviso di errore irreversibile: Errore "Certificato sconosciuto", come mostrato nell'immagine.



A questo punto è necessario controllare i certificati presentati dal server di hosting e cercare algoritmi di firma non supportati. È comune vedere **RSASSA-PSS** come l'algoritmo di firma

problematico:



Se un certificato della catena è firmato con RSASSA-PSS, la connessione non riesce. In questo caso l'acquisizione del pacchetto indica che la CA radice utilizza RSASSA-PSS per il proprio certificato:

```
Certificates (3906 bytes)
    Certificate Length: 1728
  Certificate: 308206bc308205a4a003020102021374000000243b805da9... (id-at-commonName=[redacted]
    signedCertificate
    algorithmIdentifier (sha256WithRSAEncryption)
      Padding: 0
      encrypted: e6230df257be9d34c0f57bc2f88c081c4186aaad092c8155...
    Certificate Length: 1114
  Certificate: 308204563082033ea003020102021316000000a93cd17d6... (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...
```

Soluzione

Per risolvere il problema, è necessario che un nuovo certificato venga rilasciato da un provider CA che utilizza solo uno dei tipi di firma SunMSCAPI supportati elencati nell'intera catena di certificati, come descritto in precedenza.

> **Nota**: per ulteriori dettagli sull'algoritmo di firma RSASSA-PSS, vedere https://pkisolutions.com/pkcs1v2-1rsassa-pss/

> **Nota:** Questo problema è registrato nel CSCve79330 difettoso

Informazioni su questa traduzione

Cisco ha tradotto questo documento utilizzando una combinazione di tecnologie automatiche e umane per offrire ai nostri utenti in tutto il mondo contenuti di supporto nella propria lingua. Si noti che anche la migliore traduzione automatica non sarà mai accurata come quella fornita da un traduttore professionista. Cisco Systems, Inc. non si assume alcuna responsabilità per l’accuratezza di queste traduzioni e consiglia di consultare sempre il documento originale in inglese (disponibile al link fornito).