Convalidare i Webhook di Cisco Intersight: Guida basata sulla logica

Sommario

Introduzione

In questo documento viene descritto come convalidare un webhook in intersight.

Prerequisiti

Quando Cisco Intersight invia un webhook all'applicazione, in pratica invia un evento (come un allarme server). Ma come fai a sapere che il messaggio in realtà è arrivato da Cisco e non è stato inviato da qualcun altro che ha cercato di innescare un'azione falsa nel tuo sistema?

Per risolvere questo problema, Intersight utilizza un Webhook Secret. Pensatela come un sigillo su una busta: se il sigillo è rotto o ha un aspetto diverso da quello previsto, la lettera non è considerata attendibile.

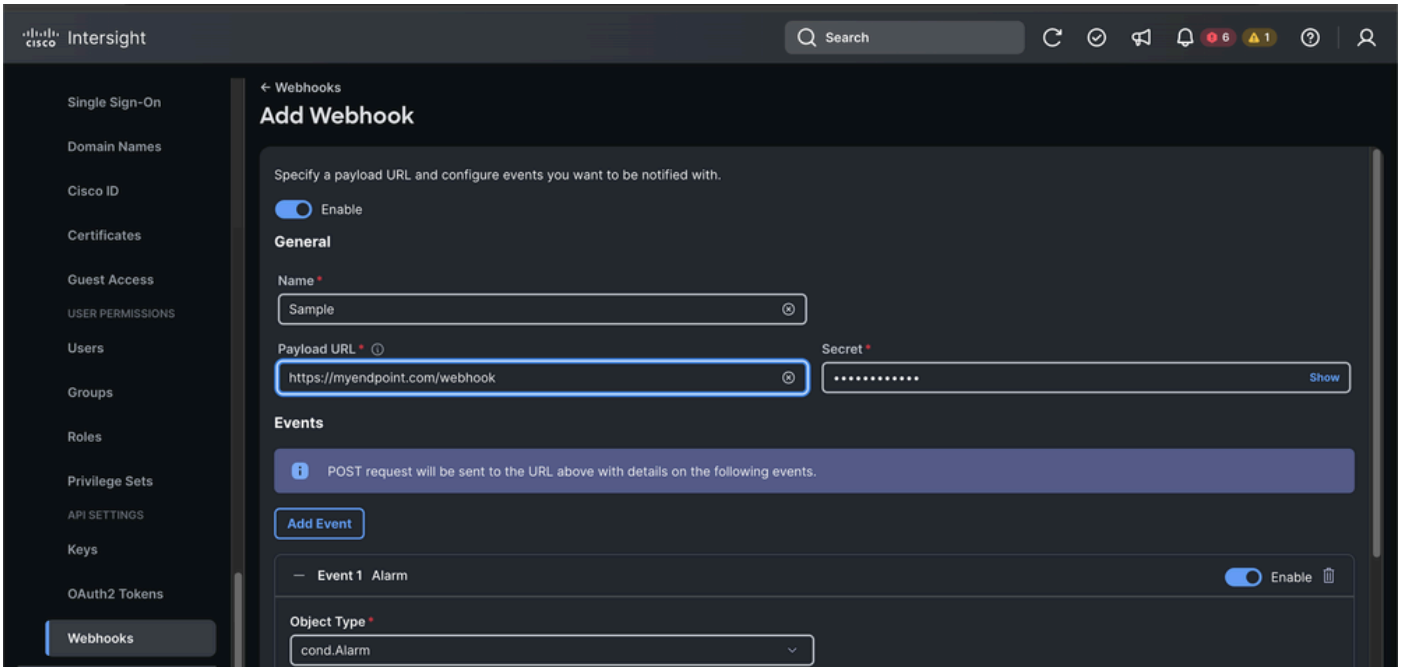
Impostazione del segreto

Il primo passo è configurare il Webhook in Intersight e stabilire un segreto condiviso.

1. Accedere a Cisco Intersight.
2. Selezionare Impostazioni > Webhook.
3. Quando si crea o si modifica un Webhook, è possibile visualizzare un campo denominato

Segreto.

4. Definire questa stringa manualmente, ad esempio secret. Una volta salvato, Intersight lo usa per firmare ogni messaggio che invia.
5. Importante: salva il segreto in modo sicuro e non divulgarlo pubblicamente.



Logica di convalida

Passaggio 1: Verifica del sigillo (il digest del corpo)

La prima cosa da controllare è se il corpo del messaggio è stato cambiato durante il viaggio. Lo facciamo usando un Hash (in particolare SHA-256).

Che cos'è un hash?

È come un'impronta digitale. Anche se si modifica una virgola in un documento di 10 pagine, l'impronta digitale cambia completamente.

La logica:

1. Prendere il corpo della richiesta raw (il testo JSON esattamente come è arrivato).
2. Eseguirlo tramite la funzione di hashing SHA-256.
3. Converte l'impronta digitale binaria in una stringa leggibile utilizzando la codifica Base64.
4. Confrontare il risultato con il Digestheader inviato da Intersight.
5. Deve essere simile a quanto riportato di seguito:SHA-256=stringa_calcolata.

Passaggio 2: Preparare la destinazione della richiesta

Intersight include la destinazione del messaggio nella sua firma per impedire attacchi di tipo replay (in cui qualcuno ruba un messaggio valido e lo invia a un endpoint diverso).

Logica:creare una stringa che combina il metodo HTTP e il percorso.

Formato:(request-target): post /your/endpoint/path

Passaggio 3: Creare la stringa di firma

Devono essere seguite in ordine rigoroso.

È qui che la maggior parte degli sviluppatori si trova nei guai. Intersight è estremamente rigoroso per quanto riguarda l'ordine, la distinzione tra maiuscole e minuscole e la formattazione delle intestazioni utilizzate per la firma. È necessario creare un singolo blocco di testo in cui ogni riga è nome-intestazione: valore.

Ordine esatto richiesto:

1. (richiesta-destinazione) (dal passaggio 2)
2. host
3. data
4. digest (valore completo dell'intestazione del digest dal passaggio 1)
5. content-type
6. content-length

```
(request-target): post /api/webhook
host: myapp.example.com
date: Mon, 09 Mar 2026 12:50:29 GMT
digest: SHA-256=L6Y...
content-type: application/json
content-length: 542
```

Passaggio 4: Genera la firma (HMAC)

A questo punto si utilizza la chiave segreta (dall'interfaccia utente di Intersight) per firmare la stringa appena creata. Utilizziamo un metodo chiamato HMAC-SHA256.

Che cos'è HMAC?

È un modo per firmare un messaggio utilizzando una chiave segreta. Solo un utente con la stessa chiave segreta può generare la stessa firma.

La logica:

1. Input:la stringa di firma del passaggio 3.
2. Chiave: Il Segreto Del Webhook.
3. Processo:eseguire l'algoritmo HMAC-SHA256.
4. Output:accetta i dati binari risultanti e Codifica Base64.

Passaggio 5: Il confronto finale

Intersight invia un'intestazione Authorization. È necessario ricostruire l'aspetto previsto dell'intestazione utilizzando la firma appena generata.

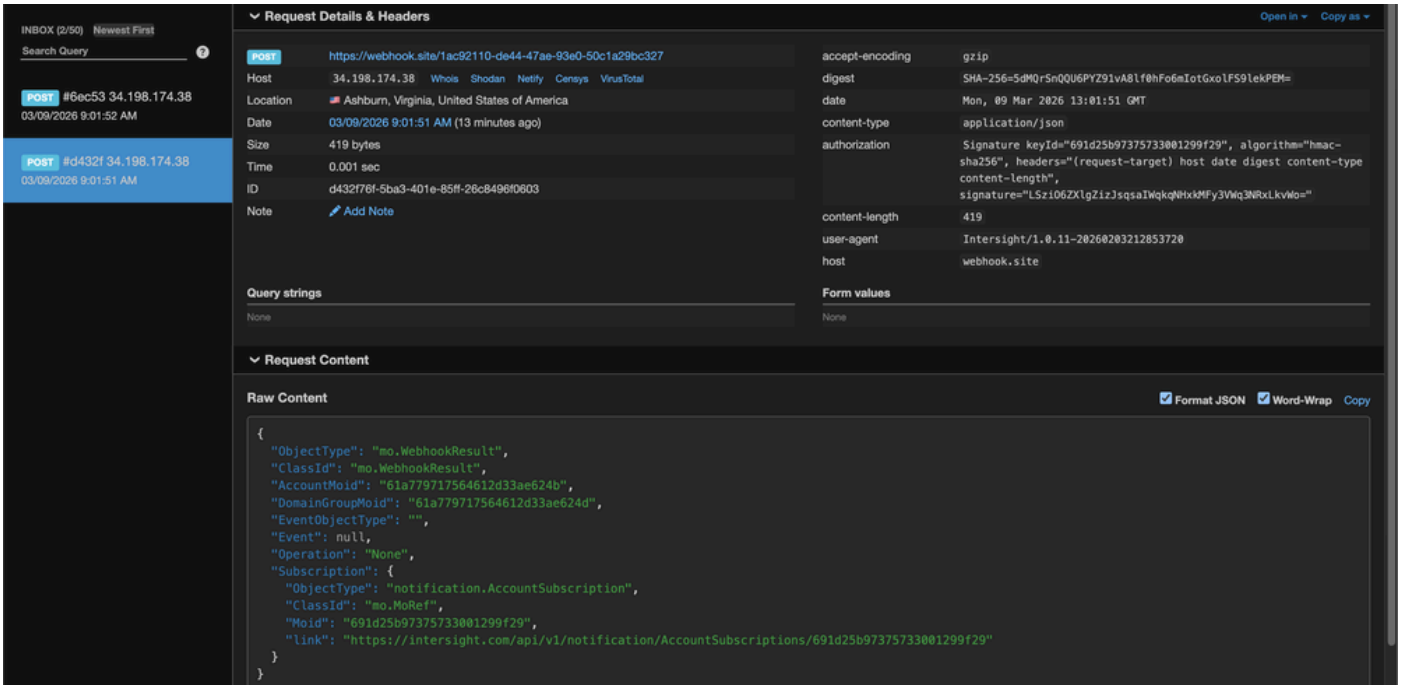
Se la stringa calcolata corrisponde all'intestazione Authorization fornita nella richiesta, il messaggio è autentico.

Considerazioni importanti

1. Inclinazione orologio: Controllare sempre l'intestazione della data. Se la richiesta ha più di 5 minuti rispetto al tempo del server, rifiutarla per evitare attacchi di tipo replay.
2. Corpo non elaborato: Non analizzare il JSON e quindi rieseguirlo prima di convalidare il digest. Librerie diverse aggiungono spaziature diverse che interrompono l'hash.
3. Ordine intestazione: Intersight convalida la firma in base all'ordine definito nella sezione headers="..." dell'intestazione Authorization. Assicurarsi che la stringa da firmare corrisponda esattamente a tale ordine.

Esempi verificabili

Per aiutarti a testare il codice, di seguito è riportato un esempio basato su un payload webhook reale inviato da Intersight.



Parametri di test

<#root>

Secret

:secret

Host

:webhook.site

Path:

/1ac92110-de44-47ae-93e0-50c1a29bc327

Date

:Mon, 09 Mar 2026 13:01:51 GMT

Content-Length

:419

Payload

:{"ObjectType":"mo.WebhookResult","ClassId":"mo.WebhookResult","AccountMoid":"61a779717564612d33ae624b"

Expected Signature

:LSziO6ZXlgZizJsqsaIWqkqNHxkMFy3VWq3NRxLkvWo=

Verifica Bash e OpenSSL

```
#!/bin/bash

# 1. Setup the inputs
SECRET="secret"
EXPECTED_SIG="LSziO6ZXlgZizJsqsaIWqkqNHxkMFy3VWq3NRxLkvWo="
PAYLOAD='{"ObjectType":"mo.WebhookResult","ClassId":"mo.WebhookResult","AccountMoid":"61a779717564612d3

# 2. Calculate the Body Digest
# We use echo -n to ensure no trailing newline is added to the payload
DIGEST=$(echo -n "$PAYLOAD" | openssl dgst -sha256 -binary | openssl base64)
FULL_DIGEST="SHA-256=$DIGEST"

# 3. Build the Signing String (Strict Order!)
# Note: The format must be exactly: header: value\n
SIGNING_STR="(request-target): post /1ac92110-de44-47ae-93e0-50c1a29bc327
host: webhook.site
date: Mon, 09 Mar 2026 13:01:51 GMT
digest: $FULL_DIGEST
content-type: application/json
content-length: 419"

# 4. Generate the HMAC-SHA256 Signature
CALCULATED_SIG=$(echo -n "$SIGNING_STR" | openssl dgst -sha256 -hmac "$SECRET" -binary | openssl base64

# 5. Output the results for comparison
echo "--- Verification Results ---"
echo "Expected Signature:   $EXPECTED_SIG"
echo "Calculated Signature: $CALCULATED_SIG"

if [ "$EXPECTED_SIG" == "$CALCULATED_SIG" ]; then
    echo "SUCCESS: The signatures match!"
else
    echo "FAILURE: The signatures do not match."
```

```
fi
```

Verifica di PowerShell

```
# 1. Setup the inputs
$Secret = "secret"
$ExpectedDigest = "5dMQrSnQQU6PYZ91vA8lf0hFo6mIotGxolFS9lekPEM="
$ExpectedSig    = "LSziO6ZXlgZizJsqsaIWqkqNHxkMFy3VWq3NRxLkvWo="

$Payload = '{"ObjectType":"mo.WebhookResult","ClassId":"mo.WebhookResult","AccountMoid":"61a77971756461

# 2. Calculate the Body Digest
$Sha256 = [System.Security.Cryptography.SHA256]::Create()
$PayloadBytes = [System.Text.Encoding]::UTF8.GetBytes($Payload)
$HashBytes = $Sha256.ComputeHash($PayloadBytes)
$CalculatedDigest = [Convert]::ToBase64String($HashBytes)

# 3. Build the Signing String (Strict Order!)
# Note: `n is the PowerShell newline character.
# The string must match the order in the Authorization header exactly.
$SigningStr = "(request-target): post /1ac92110-de44-47ae-93e0-50c1a29bc327`n" +
              "host: webhook.site`n" +
              "date: Mon, 09 Mar 2026 13:01:51 GMT`n" +
              "digest: SHA-256=$CalculatedDigest`n" +
              "content-type: application/json`n" +
              "content-length: 419"

# 4. Generate the HMAC-SHA256 Signature
$Hmac = New-Object System.Security.Cryptography.HMACSHA256
$Hmac.Key = [System.Text.Encoding]::UTF8.GetBytes($Secret)
$SigBytes = $Hmac.ComputeHash([System.Text.Encoding]::UTF8.GetBytes($SigningStr))
$CalculatedSig = [Convert]::ToBase64String($SigBytes)

# 5. Output the results for comparison
Write-Host "--- Verification Results ---" -ForegroundColor Cyan

Write-Host "Digest Match: " -NoNewline
if ($CalculatedDigest -eq $ExpectedDigest) {
    Write-Host "SUCCESS" -ForegroundColor Green
} else {
    Write-Host "FAILED" -ForegroundColor Red
}

Write-Host "Expected Signature:   $ExpectedSig"
Write-Host "Calculated Signature: $CalculatedSig"

if ($CalculatedSig -eq $ExpectedSig) {
    Write-Host "SUCCESS: The signatures match!" -ForegroundColor Green
} else {
    Write-Host "FAILURE: The signatures do not match." -ForegroundColor Red
}
```

Informazioni correlate

- Richiama richiesta API Web
- Configurazione di Cisco Intersight Webhook
- webhook/Endpoints

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).