Valide los webhooks de Cisco Intersight: Guía basada en la lógica

Contenido

Introducción

Este documento describe cómo validar un webhook en intersight.

Prerequisites

Cuando Cisco Intersight envía un webhook a la aplicación, se trata básicamente de enviar un evento (como una alarma de servidor). Pero, ¿cómo sabe que el mensaje en realidad vino de Cisco y no fue enviado por otra persona que intenta desencadenar una acción falsa en su sistema?

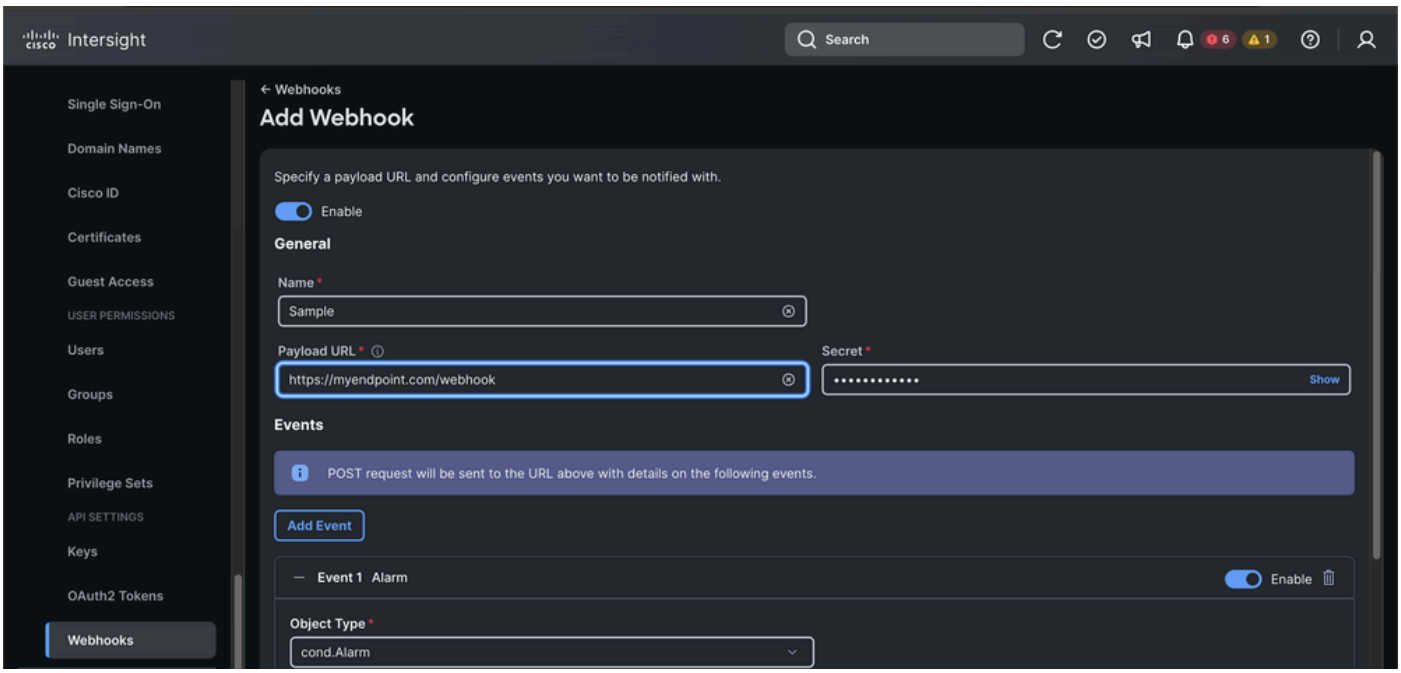
Para resolver esto, Intersight utiliza un secreto Webhook. Piense en ello como un sello en un sobre: si el sello está roto o tiene un aspecto diferente al esperado, no confía en la letra.

Configuración del secreto

El primer paso es configurar el Webhook en Intersight y establecer un secreto compartido.

1. Inicie sesión en Cisco Intersight.
2. Vaya a Configuración > Ganchos Web.
3. Al crear o editar un webhook, puede ver un campo denominado Secreto.

4. Defina esta cadena usted mismo (por ejemplo, secreto). Una vez guardado, Intersight lo utiliza para firmar cada mensaje que envía.
5. Importante: guarde este secreto de forma segura y no lo comparta públicamente.



La lógica de validación

Paso 1: Compruebe el retén (el resumen de la carrocería)

Lo primero que hay que comprobar es si el cuerpo del mensaje se cambió durante el viaje. Para ello, utilizamos un Hash (específicamente SHA-256).

¿Qué es un Hash?

Es como una huella digital. Incluso si cambia una coma en un documento de 10 páginas, la huella digital cambia por completo.

La lógica:

1. Tome el Cuerpo de solicitud sin procesar (el texto JSON exactamente como llegó).
2. Ejecute la función de hashing aSHA-256.
3. Convierta esa huella digital binaria en una cadena legible utilizandoBase64 Encoding.
4. Compare su resultado con el encabezado Digestheader enviado por Intersight.
5. Debe ser como éste:SHA-256=your_Calculation_string.

Paso 2: Preparación del destino de la solicitud

La intercepción incluye el destino del mensaje en su firma para evitar ataques de repetición (cuando alguien roba un mensaje válido y lo envía a un punto final diferente).

Logic:Create una cadena que combina el método HTTP y la ruta de acceso.

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

Paso 3: Crear la cadena de firma

Debe ser seguido en orden estricto.

Aquí es donde la mayoría de los desarrolladores se meten en problemas. La interacción es extremadamente estricta en cuanto al orden, la distinción entre mayúsculas y minúsculas y el formato de los encabezados utilizados para la firma. Debe construir un solo bloque de texto donde cada línea sea header-name: valor.

El pedido exacto requerido:

1. (request-target) (desde el paso 2)
2. anfitrión
3. fecha
4. digest (valor completo del encabezado Digest del paso 1)
5. Tipo de contenido
6. longitud del contenido

```
(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
```

Paso 4: Generar la firma (HMAC)

Ahora utiliza la Clave Secreta (de la interfaz de usuario de Intersight) para firmar la cadena que acabamos de crear. Utilizamos un método llamado HMAC-SHA256.

¿Qué es HMAC?

Es una forma de firmar un mensaje mediante una clave secreta. Sólo alguien con la misma clave secreta puede producir la misma firma.

La lógica:

1. Entrada:La cadena de firma del paso 3.
2. Clave:Su Secreto De Webhook.
3. Proceso:Ejecute el algoritmo HMAC-SHA256.
4. Salida:Tome los datos binarios resultantes yBase64 Encodeit.

Paso 5: La comparación final

Intersight envía un encabezado de autorización. Debe reconstruir el aspecto que espera que

tenga ese encabezado con la firma que acaba de generar.

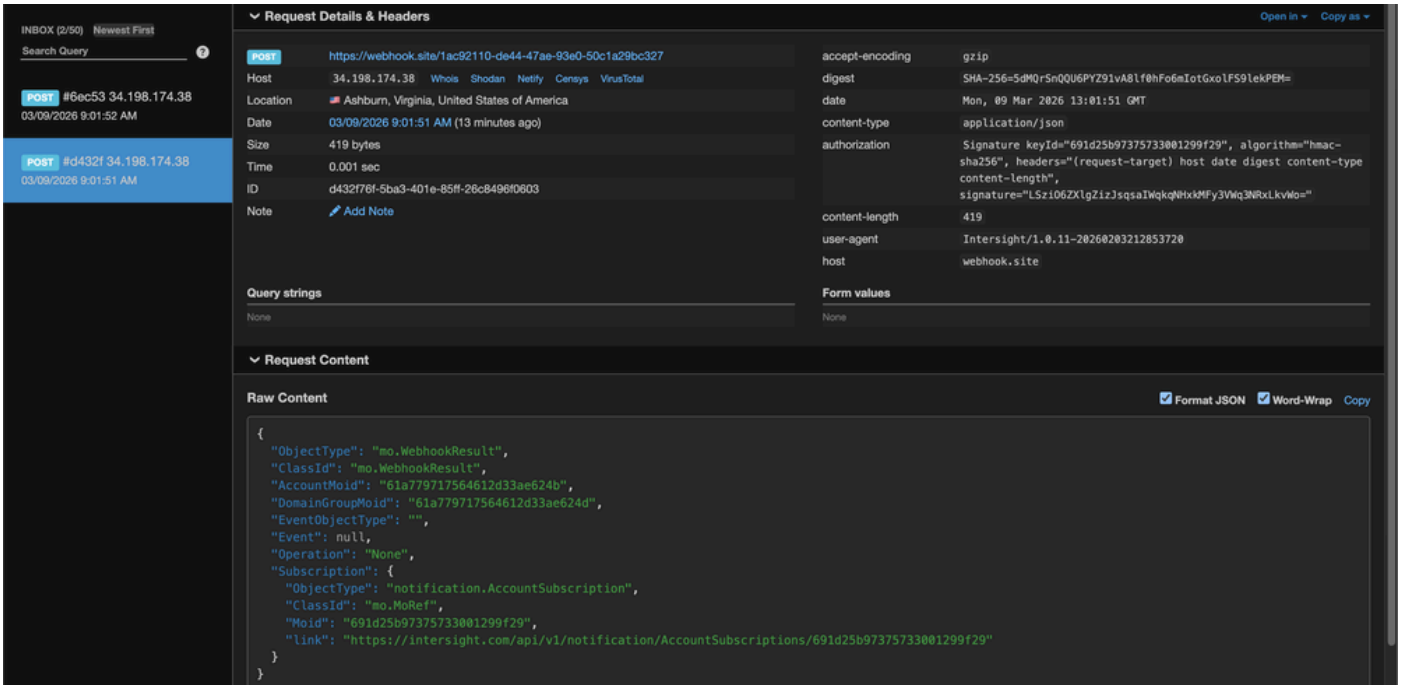
Si la cadena calculada coincide con el encabezado Authorization proporcionado en la solicitud, el mensaje es auténtico.

Consideraciones importantes

1. Desplazamiento del reloj: Compruebe siempre el encabezado de fecha. Si la solicitud tiene más de 5 minutos de antigüedad en comparación con la hora del servidor, rechácela para evitar ataques de repetición.
2. Cuerpo sin procesar: No analice el JSON y, a continuación, vuelva a consolidarlo antes de validar el resumen. Diferentes bibliotecas añaden un espaciado diferente, lo que romperá el hash.
3. Orden de encabezado: Intersight valida la firma basándose en el orden definido en la sección Headers="..."del encabezado Authorization. Asegúrese de que la cadena que se firmará coincide exactamente con ese pedido.

Ejemplos comprobables

Para ayudarle a probar su código, aquí hay un ejemplo basado en una carga útil de webhook real enviada desde Intersight.



Parámetros de prueba

<#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=

Verificación de Bash y 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
```

Verificación de 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.
$SigningStr = "(request-target): post /1ac9211O-de44-47ae-93eO-5Oc1a29bc327`n" +
              "host: webhook.site`n" +
              "date: Mon, 09 Mar 2026 13:01:51 GMT`n" +
              "digest: SHA-256=$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
}
```

Información Relacionada

- Invocar solicitud de API web
- Configuración de Webhook de Cisco Intersight
- webhook/terminales

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).

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).