Valide os webhooks da Cisco Intersight: Guia baseado em lógica

Contents

Introdução

Este documento descreve como validar um webhook no intersight.

Pré-requisitos

Quando a Cisco Intersight envia um webhook para o seu aplicativo, ele essencialmente envia um evento (como um alarme de servidor). Mas como você sabe que a mensagem realmente veio da Cisco e não foi enviada por outra pessoa tentando disparar uma ação falsa em seu sistema?

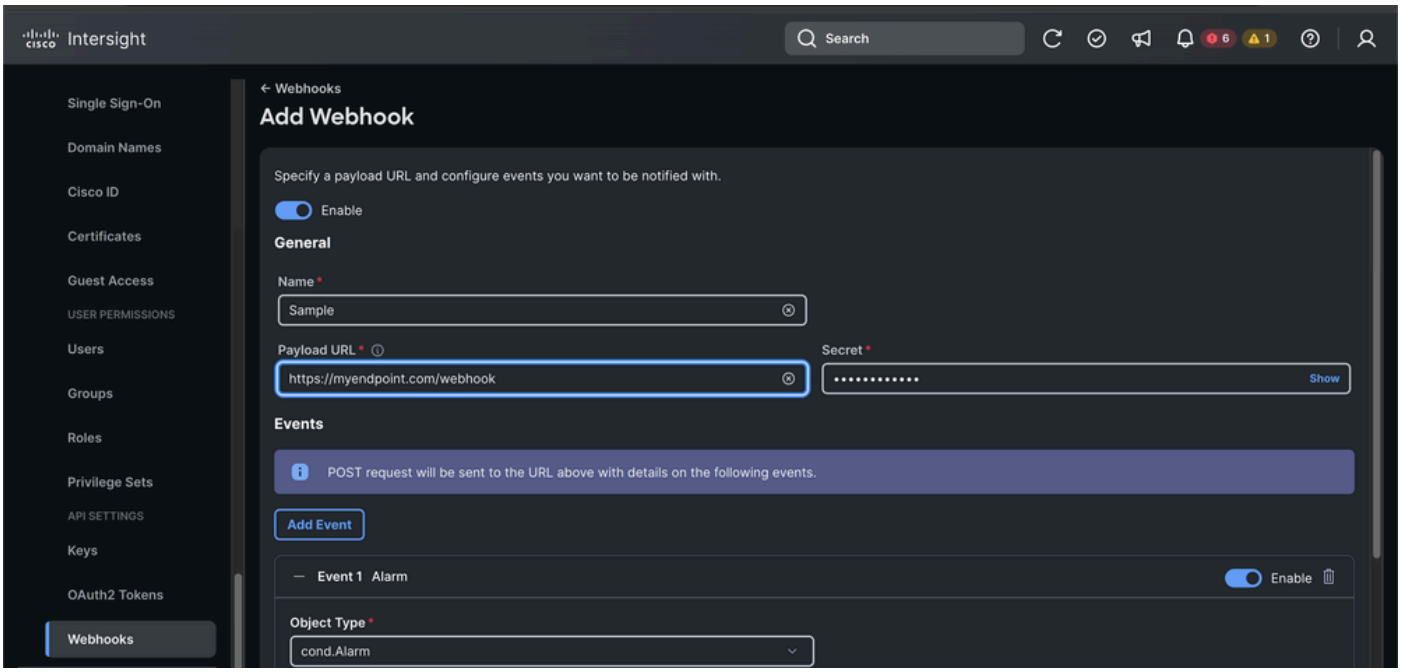
Para resolver isso, a Intersight usa um Segredo Webhook. Pense nisso como um selo em um envelope: se o selo estiver quebrado ou parecer diferente do esperado, você não confia na letra.

Definindo o segredo

A primeira etapa é configurar o Webhook no Intersight e estabelecer um Shared Secret.

1. Faça login naCisco Intersight.
2. Navegue atéSettings > Webhooks.
3. Ao criar ou editar um Webhook, você poderá ver um campo rotulado como Segredo.
4. Defina você mesmo esta string (por exemplo, ecret). Depois de salva, a Intersight usa isso

para assinar cada mensagem que envia.
5. Importante:Salve esse segredo de maneira segura e não o compartilhe publicamente.



A lógica de validação

Passo 1: Verifique o selo (o resumo do corpo)

A primeira coisa a verificar é se o corpo da mensagem foi alterado durante sua jornada. Fazemos isso usando um Hash (especificamente SHA-256).

O que é um Hash?

É como uma impressão digital. Mesmo se você alterar uma vírgula em um documento de 10 páginas, a impressão digital será completamente alterada.

A Lógica:

1. Tome oCorpo de Solicitação Bruto(o texto JSON exatamente como ele chegou).
2. Execute-o através da função de hash aSHA-256.
3. Converta essa impressão digital binária em uma cadeia de caracteres legível usando a Codificação Base64.
4. Compare seu resultado com o Cabeçalho de Digestão enviado pela Intersight.
5. Ele deve ser assim:SHA-256=sua_string_calculada.

Passo 2: Preparar o Alvo da Solicitação

O Intersight inclui o destino da mensagem em sua assinatura para evitar ataques de repetição (quando alguém rouba uma mensagem válida e a envia para um endpoint diferente).

A Lógica:Crie uma string que combine o método HTTP e o caminho.

Formato:(request-target): postar /seu/ponto final/caminho

Passo 3: Criar a Cadeia de Caracteres de Assinatura

Deve ser seguido em ordem estrita.

É aqui que a maioria dos desenvolvedores se deparam com problemas. A Intersight é extremamente rígida sobre a ordem, a distinção entre maiúsculas e minúsculas e a formatação dos cabeçalhos usados para a assinatura. Você deve criar um único bloco de texto onde cada linha é nome de cabeçalho: valor.

A ordem exata necessária:

1. (solicitação-destino) (Da Etapa 2)
2. host
3. data
4. digest (O valor completo do cabeçalho Digest da Etapa 1)
5. Tipo de conteúdo
6. comprimento de conteúdo

```
(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
```

Passo 4: Gerar a assinatura (o HMAC)

Agora você usa a chave secreta (da interface do usuário da Intersight) para assinar a sequência que acabamos de criar. Usamos um método chamado HMAC-SHA256.

O que é HMAC?

É uma maneira de assinar uma mensagem usando uma chave secreta. Somente alguém com a mesma chave secreta pode produzir a mesma assinatura.

A Lógica:

1. Entrada:A string de assinatura da etapa 3.
2. Key:Seu Segredo De Webhook.
3. Processo:Executar o algoritmo HMAC-SHA256.
4. Saída:Pegue os dados binários resultantes eBase64 Encodeit.

Passo 5: A comparação final

A Intersight envia um cabeçalho Authorization. Você precisa reconstruir a aparência esperada

desse cabeçalho usando a assinatura que acabou de gerar.

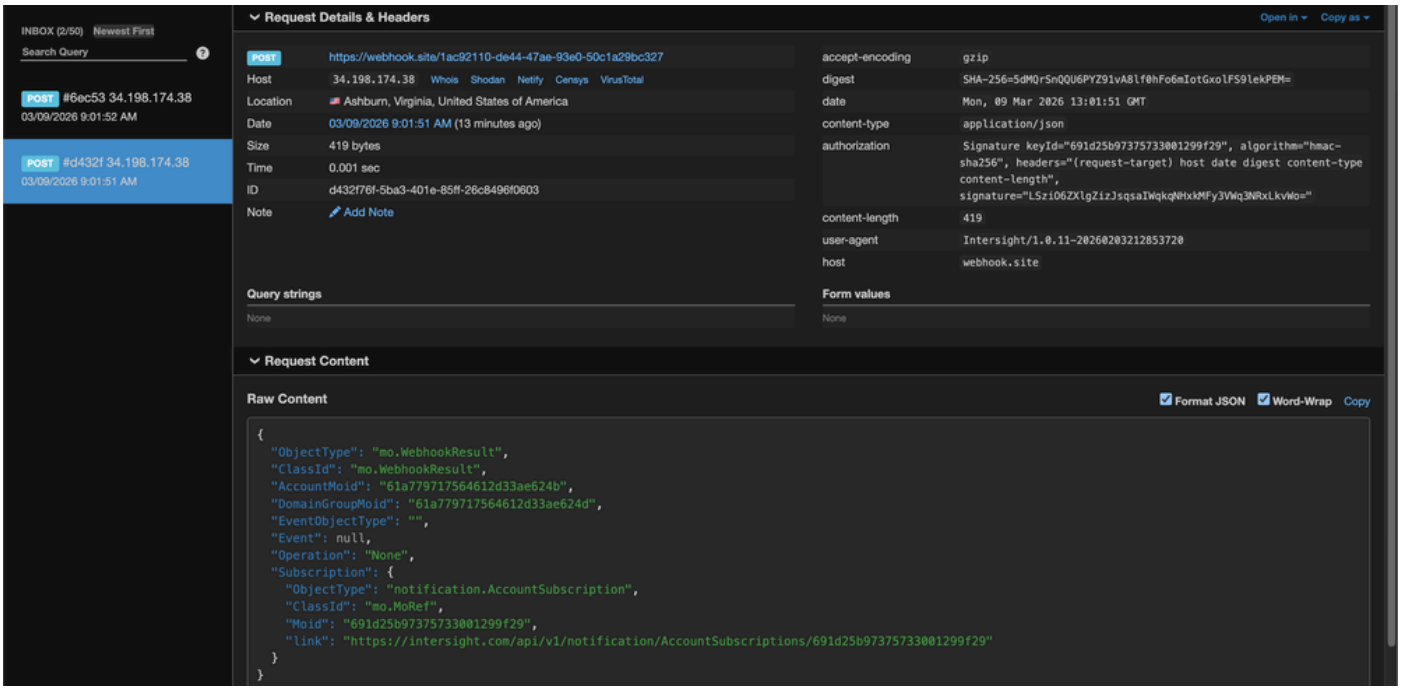
Se sua string calculada corresponder ao cabeçalho Autorização fornecido na solicitação, a mensagem será autêntica.

Considerações importantes

1. Distorção do relógio: Sempre verifique o cabeçalho da data. Se a solicitação tiver mais de 5 minutos em comparação ao horário do servidor, rejeite-a para evitar ataques de repetição.
2. Corpo Bruto: Não analise o JSON e, em seguida, retorne-o antes de validar o resumo. Bibliotecas diferentes adicionam espaçamento diferente, o que quebrará o hash.
3. Ordem do cabeçalho: A Intersight valida a assinatura com base na ordem definida na seção theheaders="..." do cabeçalhoAutorização. Certifique-se de que sua string a assinar corresponda exatamente a essa ordem.

Exemplos verificáveis

Para ajudá-lo a testar seu código, aqui está um exemplo baseado em uma carga útil real de webhook enviada da Intersight.



Parâmetros de teste

<#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=

Verificação Bash & 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
```

Verificação do PowerShell

```
# 1. Setup the inputs
$Secret = "secret"
$ExpectedDigest = "5dMQrSnQQU6PYZ91vA8lfOhFo6mIotGxolFS9lekPEM="
$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
}
```

Informações Relacionadas

- Chamar Solicitação de API Web

- Configuração do Cisco Intersight Webhook
- webhook/Endpoints

Sobre esta tradução
A Cisco traduziu este documento com a ajuda de tecnologias de tradução automática e humana para oferecer conteúdo de suporte aos seus usuários no seu próprio idioma, independentemente da localização.

Observe que mesmo a melhor tradução automática não será tão precisa quanto as realizadas por um tradutor profissional.

A Cisco Systems, Inc. não se responsabiliza pela precisão destas traduções e recomenda que o documento original em inglês (link fornecido) seja sempre consultado.