Cisco Intersight Webhooks를 검증합니다. 논리 기반 설명서

목차

- 소개
- 사전 요구 사항
- 비밀 설정
- 검증 로직
  - 1단계: 봉인 확인(Body Digest)
  - 2단계: 요청 대상 준비
  - 3단계: 서명 문자열 만들기
  - 4단계: 서명 생성(HMAC)
  - 5단계: 최종 비교
- 중요 고려 사항
- 확인 가능한 예
  - 테스트 매개 변수
  - Bash 및 OpenSSL 확인
  - PowerShell 확인
- 관련 정보

소개

이 문서에서는 Intersight에서 Webhook을 확인하는 방법에 대해 설명합니다.

사전 요구 사항

Cisco Intersight에서 웹 훅을 애플리케이션에 전송하면 기본적으로 이벤트(예: 서버 경보)가 전송됩니다. 그러나 이 메시지가 실제로 Cisco에서 왔고 귀하의 시스템에서 가짜 작업을 시작하려는 다른 사람이 보낸 것이 아니라는 것을 어떻게 알 수 있습니까?

이를 해결하기 위해 Intersight는 Webhook Secret을 사용합니다. 봉투에 든 물개처럼 생각해보세요. 도장이 깨지거나 예상과 다르게 보이면 편지를 믿지 않는다.

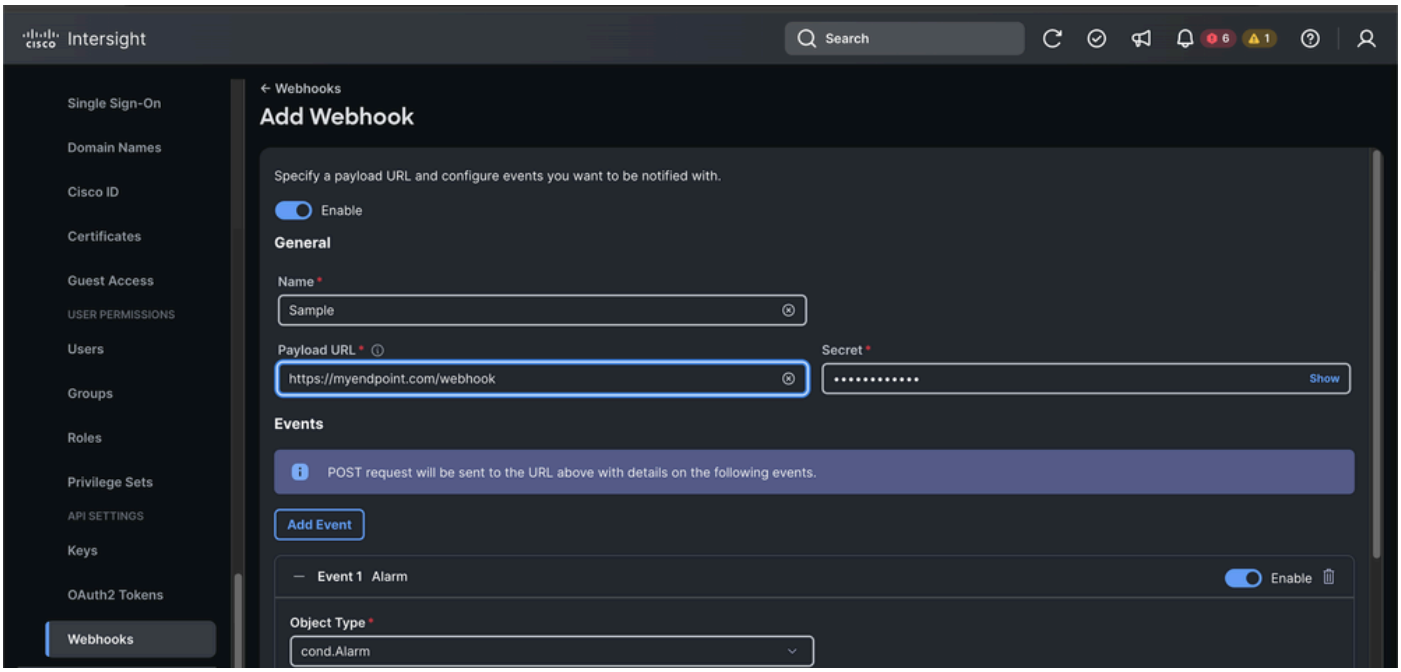
비밀 설정

첫 번째 단계는 Intersight에서 Webhook을 구성하고 공유 암호를 설정하는 것입니다.

1. Cisco Intersight에 로그인합니다.
2. 설정 > Webhooks로 이동합니다.
3. Webhook을 생성하거나 편집할 때 Secret이라는 필드가 표시됩니다.
4. 이 문자열을 직접 정의합니다(예: select). 저장되면 Intersight는 이를 사용하여 보내는 모든 메

시지에 서명합니다.

5. 중요:이 비밀을 안전하게 저장하고 공개적으로 공유하지 않습니다.



검증 로직

1단계: 봉인 확인(Body Digest)

가장 먼저 확인해야 할 것은 메시지 본문이 이동 중에 변경되었는지 여부입니다. 여기서는 해시(특히 SHA-256)를 사용합니다.

Hash란?

그것은 마치 지문과도 같습니다. 10페이지 분량의 문서에서 쉼표 하나를 변경해도 지문이 완전히 변경됩니다.

논리:

1. Raw Request Body(도착한 JSON 텍스트)를 가져옵니다.
2. ShaSHA-256 해싱 기능을 실행합니다.
3. Base64 Encoding을 사용하여 이진 핑거프린트를 읽기 가능한 문자열로 변환합니다.
4. 결과를 Intersight에서 보낸 Digespeader와 비교합니다.
5. 다음과 같아야 합니다. SHA-256=your_calculated_string.

2단계: 요청 대상 준비

Intersight는 재생 공격을 방지하기 위해 메시지의 대상을 시그니처에 포함합니다(누군가가 유효한 메시지를 훔쳐내어 다른 엔드포인트로 전송하는 경우).

논리:HTTP 메서드와 경로를 결합하는 문자열을 만듭니다.

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

3단계: 서명 문자열 만들기

엄격한 순서대로 따라야 합니다.

대부분의 개발자가 이 부분에서 어려움을 겪습니다. Intersight는 서명에 사용되는 헤더의 순서, 대/소문자 구분 및 형식에 대해 매우 엄격합니다. 각 행이 header-name인 텍스트 블록을 작성해야 합니다. 가치.

정확한 주문:

1. (request-target)(2단계)
2. 호스트
3. 날짜
4. digest(1단계의 다이제스트 헤더의 전체 값)
5. content-type
6. 콘텐츠 길이

```
(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
```

4단계: 서명 생성(HMAC)

이제 Intersight UI에서 Secret Key(비밀 키)를 사용하여 방금 만든 문자열에 서명합니다. 우리는 HMAC-SHA256이라는 방법을 사용합니다.

HMAC란 무엇입니까?

비밀 키를 사용하여 메시지에 서명하는 방법입니다. 비밀키가 같은 사람만이 같은 서명을 만들 수 있다.

논리:

1. 입력: 3단계의 서명 문자열입니다.
2. 키:Webhook Secret.
3. 프로세스: HMAC-SHA256 알고리즘을 실행합니다.
4. 출력:결과 이진 데이터 및 Base64 Encodeit를 가져옵니다.

5단계: 최종 비교

Intersight에서 Authorization 헤더를 전송합니다. 방금 생성한 시그니처를 사용하여 헤더가 어떤 모양이 될 것으로 예상하는지를 재구성해야 합니다.

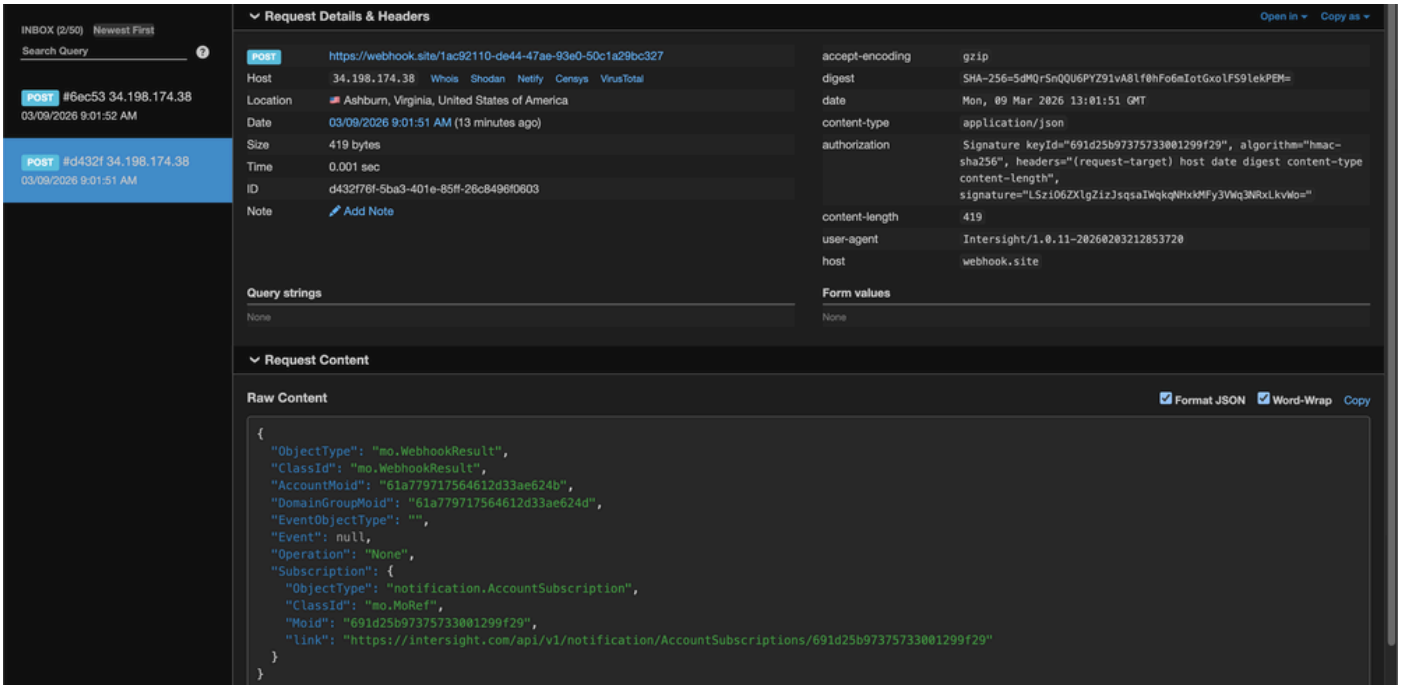
계산된 문자열이 요청에서 제공된 Authorization 헤더와 일치하면 메시지는 정품입니다.

중요 고려 사항

1. 클럭 기울이기: 항상 날짜 머리글을 확인합니다. 요청이 서버 시간과 비교하여 5분 이상 지난 경우 요청을 거부하여 재생 공격을 방지합니다.
2. 원시 본문: 다이제스트를 검증하기 전에 JSON을 구문 분석한 다음 다시 분류하지 마십시오. 라이브러리마다 다른 간격이 추가되어 해시가 중단됩니다.
3. 헤더 순서: Intersight는 Authorization 헤더의 headers="..."섹션에 정의된 순서에 따라 서명을 검증합니다. String to Sign(서명할 문자열)이 해당 순서와 정확히 일치하는지 확인합니다.

확인 가능한 예

코드를 테스트하는 데 도움이 되도록 Intersight에서 보낸 실제 webhook 페이로드를 기반으로 한 예제가 있습니다.



테스트 매개 변수

<#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=

Bash 및 OpenSSL 확인

```
#!/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
```

PowerShell 확인

```
# 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
}
```

관련 정보

- 웹 API 요청 호출
- Cisco Intersight Webhook 컨피그레이션
- 웹후크/엔드포인트

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.