

De Cisco Intersight Webhooks valideren: op logica gebaseerde gids

Inhoud

[Inleiding](#)

[Voorwaarden](#)

[Het geheim instellen](#)

[De validatielogica](#)

[Stap 1: Controleer het zegel \(The Body Digest\)](#)

[Stap 2: Bereid het aanvraagdoel voor](#)

[Stap 3: Maak de tekenreeks aan](#)

[Stap 4: Genereer de handtekening \(de HMAC\)](#)

[Stap 5: De laatste vergelijking](#)

[Belangrijke overwegingen](#)

[Verifieerbare voorbeelden](#)

[Testparameters](#)

[Verificatie via Bash en OpenSSL](#)

[PowerShell-verificatie](#)

[Gerelateerde informatie](#)

Inleiding

In dit document wordt beschreven hoe u een webhook in Intersight kunt valideren.

Voorwaarden

Wanneer Cisco Intersight een webhook naar uw toepassing verzendt, wordt in wezen een gebeurtenis verzonden (zoals een serveralarm). Maar hoe weet je dat het bericht eigenlijk van Cisco kwam en niet werd verzonden door iemand anders die een nepactie in je systeem probeerde te activeren?

Om dit op te lossen maakt Intersight gebruik van een Webhook Secret. Zie het als een zegel op een envelop: als het zegel is verbroken of er anders uitziet dan verwacht, vertrouw je de brief niet.

Het geheim instellen

De eerste stap is om de Webhook in Intersight te configureren en een Gedeeld Geheim vast te stellen.

1. Meld u aan bij Cisco Intersight.
2. Ga naar Instellingen > Webhooks.

3. Wanneer u een Webhook maakt of bewerkt, ziet u een veld met de naam Secret.
4. Definieer deze string zelf (bijvoorbeeld geheim). Eenmaal opgeslagen, gebruikt Intersight dit om elk bericht dat het verzendt te ondertekenen.
5. Belangrijk: bewaar dit geheim op een veilige manier en deel het niet publiekelijk.

The screenshot shows the Cisco Intersight interface for adding a webhook. The sidebar on the left contains various navigation links. The main panel is titled 'Add Webhook' and includes a toggle to 'Enable' the webhook. Under the 'General' section, there are fields for 'Name' (set to 'Sample') and 'Payload URL' (set to 'https://myendpoint.com/webhook'). A 'Secret' field is also present, currently masked with dots and a 'Show' button. Below the general settings is an 'Events' section with an 'Add Event' button. A table below shows 'Event 1 Alarm' with an 'Object Type' dropdown set to 'cond.Alarm' and an 'Enable' toggle.

De validatielogica

Stap 1: Controleer het zegel (The Body Digest)

Het eerste wat u moet controleren is of het berichtlichaam tijdens de reis is gewijzigd. Dit doen we met behulp van een Hash (met name SHA-256).

Wat is een hash?

Het is als een vingerafdruk. Zelfs als u één komma in een document van 10 pagina's wijzigt, verandert de vingerafdruk volledig.

De logica:

1. Neem de Raw Request Body (de JSON-tekst precies zoals deze is aangekomen).
2. Voer het uit via de aSHA-256-hashingfunctie.
3. Converteer die binaire vingerafdruk naar een leesbare tekenreeks met Base64 Encoding.
4. Vergelijk uw resultaat met de Digestheader verzonden door Intersight.
5. Het moet als volgt zijn: SHA-256=your_calculated_string.

Stap 2: Bereid het aanvraagdoel voor

Intersight omvat de bestemming van het bericht in de handtekening om replay-aanvallen te voorkomen (waarbij iemand een geldig bericht steelt en naar een ander eindpunt stuurt).

De logica: maak een tekenreeks die de HTTP-methode en het pad combineert.

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

Stap 3: Maak de tekenreeks aan

Moet strikt worden gevolgd.

Dit is waar de meeste ontwikkelaars in de problemen komen. Intersight is uiterst strikt over de volgorde, hoofdlettergevoeligheid en opmaak van de headers die voor de handtekening worden gebruikt. U moet één tekstblok maken waarin elke regel header-naam: waarde is.

De exacte volgorde is vereist:

1. (request-target) (vanaf stap 2)
2. gastheer
3. datum
4. digest (De volledige waarde van de header Digest uit stap 1)
5. content-type
6. inhoudslengte

```
(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
```

Stap 4: Genereer de handtekening (de HMAC)

Nu gebruik je de geheime sleutel (van de Intersight UI) om de string te ondertekenen die we net hebben gebouwd. We gebruiken een methode genaamd HMAC-SHA256.

Wat is HMAC?

Het is een manier om een bericht te ondertekenen met behulp van een geheime sleutel. Alleen iemand met dezelfde geheime sleutel kan dezelfde handtekening produceren.

De logica:

1. Input: De tekenreeks uit stap 3.
2. Sleutel: Uw Webhook Secret.
3. Proces: Voer het HMAC-SHA256-algoritme uit.
4. Uitvoer: neem de resulterende binaire gegevens en Base64 Encodeit.

Stap 5: De laatste vergelijking

Intersight verzendt een autorisatie header. U moet reconstrueren wat u verwacht dat die koptekst eruit ziet met behulp van de handtekening die u zojuist hebt gegenereerd.

Als de berekende tekenreeks overeenkomt met de autorisatie-header die in de aanvraag is opgegeven, is het bericht authentiek.

Belangrijke overwegingen

- 1. Clock Skew: Controleer altijd de dateheader. Als het verzoek meer dan 5 minuten oud is in vergelijking met uw servertijd, wijst u het af om replay-aanvallen te voorkomen.
- 2. Rauw lichaam: parseer de JSON niet en bevestig deze vervolgens opnieuw voordat u de digest valideert. Verschillende bibliotheken voegen verschillende spativering toe, waardoor de hash wordt verbroken.
- 3. Header Order: Intersight valideert de handtekening op basis van de volgorde die is gedefinieerd in de headers="..."sectie van de header van de autorisatie. Zorg ervoor dat je String to Sign precies overeenkomt met die volgorde.

Verifieerbare voorbeelden

Om u te helpen uw code te testen, is hier een voorbeeld op basis van een echte webhook-payload die door Intersight is verzonden.

INBOX (2/50) Newest First

Search Query

POST #6ec53 34.198.174.38 03/09/2026 9:01:52 AM

POST #d432f 34.198.174.38 03/09/2026 9:01:51 AM

Request Details & Headers

POST https://webhook.site/1ac92110-de44-47ae-93e0-50c1a29bc327

Host 34.198.174.38 Whois Shodan Netify Censys VirusTotal

Location Ashburn, Virginia, United States of America

Date 03/09/2026 9:01:51 AM (13 minutes ago)

Size 419 bytes

Time 0.001 sec

ID d432f76f-5ba3-401e-85ff-26c8496f0603

Note Add Note

accept-encoding gzip

digest SHA-256=5dM0rSnQU06PYZ91vA8lF0hF06mIot6xoLF59lekPEH=

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

content-type application/json

authorization Signature keyId="691d25b97375733001299f29", algorithm="hmac-sha256", headers="(request-target) host date digest content-type content-length", signature="LSz106ZXlgZir3sqsaIWqkqNhxkMFy3Wq3NRxLkvWo="

content-length 419

user-agent Intersight/1.0.11-20260203212853720

host webhook.site

Query strings

None

Form values

None

Request Content

Raw Content

Format JSON Word-Wrap Copy

```
{
  "ObjectType": "mo.WebhookResult",
  "ClassId": "mo.WebhookResult",
  "AccountMoid": "61a779717564612d33ae624b",
  "DomainGroupMoid": "61a779717564612d33ae624d",
  "EventObjectType": "",
  "Event": null,
  "Operation": "None",
  "Subscription": {
    "ObjectType": "notification.AccountSubscription",
    "ClassId": "mo.MoRef",
    "Moid": "691d25b97375733001299f29",
    "Link": "https://intersight.com/api/v1/notification/AccountSubscriptions/691d25b97375733001299f29"
  }
}
```

Testparameters

<#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

:LSziO6ZXlgZizJsqaIWqkqNHxkMFy3VWq3NRxLkvWo=

Verificatie via Bash en OpenSSL

```
#!/bin/bash
```

```
# 1. Setup the inputs
```

```
SECRET="secret"
```

```
EXPECTED_SIG="LSziO6ZXlgZizJsqaIWqkqNHxkMFy3VWq3NRxLkvWo="
```

```
PAYLOAD='{"ObjectType":"mo.WebhookResult","ClassId":"mo.WebhookResult","AccountMoid":"61a779717564612d33ae624b"
```

```
# 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-verificatie

```

# 1. Setup the inputs
$Secret = "secret"
$ExpectedDigest = "5dMQRsnQQU6PYZ91vA81f0hFo6mIotGxo1FS91ekPEM="
$ExpectedSig     = "LSzi06ZX1gZizJsqsaIWqkqNHxkMFy3VWq3NRxLkvWo="

$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
}

```

Gerelateerde informatie

- [Web API-verzoek oproepen](#)
- [Cisco Intersight Webhook Configuration](#)
- [webhook/eindpunten](#)

Over deze vertaling

Cisco heeft dit document vertaald via een combinatie van machine- en menselijke technologie om onze gebruikers wereldwijd ondersteuningscontent te bieden in hun eigen taal. Houd er rekening mee dat zelfs de beste machinevertaling niet net zo nauwkeurig is als die van een professionele vertaler. Cisco Systems, Inc. is niet aansprakelijk voor de nauwkeurigheid van deze vertalingen en raadt aan altijd het oorspronkelijke Engelstalige document (link) te raadplegen.