WebHook validieren

Mit dem WebHook wird folgender Header gesendet:

`HTTP_X_HUB_SIGNATURE_256`

Wenn Sie die Postdaten ermitteln und mit Ihrem Secret den Hash ermitteln, können Sie sicherstellen, dass der Aufruf tatsächlich von DeDeSales erfolgt ist.

Testen der WebHook-Validierung

Sie können die folgenden `secret` und `payload` Werte verwenden, um zu überprüfen, ob Ihre Implementierung korrekt ist:

- `secret`: `It's a Secret to Everybody`
- `payload`: `Hello, World!`

Wenn Ihre Implementierung korrekt ist, sollten die von Ihnen erstellten Signaturen mit den folgenden Signaturwerten übereinstimmen:

- Signatur: `757107ea0eb2509fc211221cce984b8a37570b6d7586c22c46f4379c8b043e17`
- X-Hub-Signatur-256: `sha256=757107ea0eb2509fc211221cce984b8a37570b6d7586c22c46f4379c8b043e17`

Beispiele

Sie können eine Programmiersprache Ihrer Wahl verwenden, um die HMAC-Verifizierung in Ihrem Code zu implementieren. Nachfolgend finden Sie einige Beispiele, die zeigen, wie eine Implementierung in verschiedenen Programmiersprachen aussehen könnte.

Ruby-Beispiel

Beispielsweise kannst du die folgende `verify_signature`-Funktion definieren:

```
def verify_signature(payload_body)
  signature = 'sha256=' + OpenSSL::HMAC.hexdigest(OpenSSL::Digest.new('sha256'), ENV['SECRET_TOKEN'],
payload_body)
  return halt 500, "Signatures didn't match!" unless Rack::Utils.secure_compare(signature,
request.env['HTTP_X_HUB_SIGNATURE_256'])
end
```

Anschließend kannst du die Funktion aufrufen, wenn du eine WebHook empfängst:

```
post '/payload' do
  request.body.rewind
  payload_body = request.body.read
  verify_signature(payload_body)
  push = JSON.parse(payload_body)
  "I got some JSON: #{push.inspect}"
end
```

Pyhton-Beispiel

Du kannst beispielsweise die folgende `verify_signature`-Funktion definieren und aufrufen, wenn du eine WebHook empfängst:

```
import hashlib
import hmac
def verify_signature(payload_body, secret_token, signature_header):
    """Verify that the payload was sent from GitHub by validating SHA256.

    Raise and return 403 if not authorized.

    Args:
        payload_body: original request body to verify (request.body())
        secret_token: GitHub app webhook token (WEBHOOK_SECRET)
        signature_header: header received from GitHub (x-hub-signature-256)
    """
    if not signature_header:
        raise HTTPException(status_code=403, detail="x-hub-signature-256 header is missing!")
    hash_object = hmac.new(secret_token.encode('utf-8'), msg=payload_body, digestmod=hashlib.sha256)
    expected_signature = "sha256=" + hash_object.hexdigest()
    if not hmac.compare_digest(expected_signature, signature_header):
        raise HTTPException(status_code=403, detail="Request signatures didn't match!")
```

JavaScript-Beispiel

Sie können beispielsweise die folgende `verifySignature`-Funktion definieren und sie in jeder JavaScript-Umgebung aufrufen, wenn Sie eine Webhook empfangen:

```
let encoder = new TextEncoder();
```

```
async function verifySignature(secret, header, payload) {
    let parts = header.split("=");
    let sigHex = parts[1];

    let algorithm = { name: "HMAC", hash: { name: 'SHA-256' } };

    let keyBytes = encoder.encode(secret);
    let extractable = false;
    let key = await crypto.subtle.importKey(
        "raw",
        keyBytes,
        algorithm,
        extractable,
        [ "sign", "verify" ],
    );

    let sigBytes = hexToBytes(sigHex);
    let dataBytes = encoder.encode(payload);
    let equal = await crypto.subtle.verify(
        algorithm.name,
        key,
        sigBytes,
        dataBytes,
    );

    return equal;
}

function hexToBytes(hex) {
    let len = hex.length / 2;
    let bytes = new Uint8Array(len);

    let index = 0;
    for (let i = 0; i < hex.length; i += 2) {
        let c = hex.slice(i, i + 2);
        let b = parseInt(c, 16);
        bytes[index] = b;
        index += 1;
    }
```

```
    return bytes;
}
```

Typescript-Beispiel

Du kannst beispielsweise die folgende `verify_signature`-Funktion definieren und aufrufen, wenn du eine WebHook empfängst:

JavaScript

```
import { Webhooks } from "@octokit/webhooks";

const webhooks = new Webhooks({
  secret: process.env.WEBHOOK_SECRET,
});

const handleWebhook = async (req, res) => {
  const signature = req.headers["x-hub-signature-256"];
  const body = await req.text();

  if (!(await webhooks.verify(body, signature))) {
    res.status(401).send("Unauthorized");
    return;
  }

  // The rest of your logic here
};
```

C#-Beispiel

Beispielsweise kannst du die folgende `verify_signature`-Funktion definieren:

```
using System;
using System.Text;
using System.Security.Cryptography;

public bool verify_signature( string header_HTTP_X_HUB_SIGNATURE_256, string postData, string secret )
{
  HMACSHA256 mySHA256 = new HMACSHA256(Encoding.ASCII.GetBytes(secret));
  byte[] hashValue = mySHA256.ComputeHash(Encoding.ASCII.GetBytes(postData));

  return header_HTTP_X_HUB_SIGNATURE_256 == PrintByteArray(hashValue);
```

```
}

private String PrintByteArray(byte[] array)
{
    String result = String.Empty;
    for (int i = 0; i < array.Length; i++)
        result += $"{array[i]:X2}";

    return result;
}
```

Revision #8
Created 4 March 2025 14:06:56 by DeDeNet GmbH
Updated 13 January 2026 11:05:57 by DeDeNet GmbH