Verifying Webhook Signatures

Ask AI

When Orbital sends a webhook (an HTTP POST to your `notifyUrl`), it also attaches a digital signature in the headers. That signature is generated using Orbital's private key, and you can verify it using Orbital's public key.

If the signature matches, you know two things:

- The webhook really came from Orbital (authenticity).
- The body hasn't been changed in transit (integrity).

If it doesn't match, reject the webhook because the request may have been forged and it isn't from Orbital.

How webhook signing works

- Orbital computes an `RSA-SHA256` signature of the raw JSON request body.
- The signature is included in the `X-Orbital-Signature` header (Base64 encoded).
- You verify the signature against Orbital's public key.
- If verification fails, you must reject the webhook and ignore its contents.

Orbital Public Key

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA3sQxyMwmDXQm3mumy345
E4NZID1guNsOYSMC7WDRbwpbCSnLDSXRZTiImdONjOTeH7UZtfF1G7YY78LrxYUS
x0dnqpAMfCgfJwt/JPMIJcfg5be2WaKMjC5uXK9Shlp9JUSFfV93YChcTkOTyv2j
CmQ+KU1P3UafpmKjCc5BwR9AIut6AGc6xtIfqb9VRa1hWGTS6XLuZQlH2Q0qYYnY
9zJVFPuBucV5t3w6brV8klpYs4OhRRKBn+2FI8bSltoreoTOdKpN5ot7mKK8uzaO
vjrHPUJbkwc0r6KUKTVqyBYz5GUAzt2ld6HYUP1g9zD6MGNOM/jEShThmDKSVm8G
DQIDAQAB
-----END PUBLIC KEY-----
```

How to Verify your Webhook Signature

1. **Get Event Data**: Orbital sends your server a webhook which contains the payload of event data along with a unique signature. The request contains the body and the metadata which has the Orbital signature.

- **Body (JSON)** - the actual event data, E.g

JSON

```
 {
  "id": "d3da3a8d-1e4b-4a24-a2e5-1da62934c169",
  "externalId": "ORBITAL-CUSTOMER-REFERENCE-0001",
  "status": "initiated",
  "type": "deposit",
  "updatedAt": "2025-09-03T09:29:31.946Z",
  "event": "crypto_payment_status_updated"
}
```

- **Headers** - metadata attached to the request including the Orbital signature

JSON

```
Content-Type: application/json
X-Orbital-Signature: AbCdEfGhIj... (base64 string)
X-Orbital-Timestamp: 2025-09-02T12:00:00Z
```

2. **Get the Orbital Public key** : This is the public key that corresponds to Orbital's private signing key. You'll need this to verify the signature. See full key above.

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQE...
-----END PUBLIC KEY-----
```

3. **Create the verifySignature Function**: Create a function named verifySignature that accepts the data, signature, and public key as its arguments. Inside your function, create a verifier object using `RSA-SHA256`. This sets the algorithm used for signing, which is similar to what Orbital uses.
4. **Run the Verification**: Finally, run the verification to check the signature against the data and public key. This method returns **true** if the signature is valid and **false** if it is not.

Below is a Typescript example of Webhook Verification

TypeScript (Node.js)

Verify Webhook

```
const crypto = require('crypto');

const data = {
  "id": "d3da3a8d-1e4b-4a24-a2e5-1da62934c169",
  "externalId": "ORBITAL-CUSTOMER-REFERENCE-0001",
  "status": "initiated",
  "type": "deposit",
  "updatedAt": "2025-09-03T09:29:31.946Z",
  "event": "crypto_payment_status_updated"
}

const signature =

'd9dnS7vz4zw6erJdvBDY2w3Yk6VJj+O/51VcJJcBaoYYDZXg8Xt1pGgTwDcDTV+ohxEOY6MmJUX7jgC22KoN7jXbj5WLSAdwyZJPlnjX1btMDneCrr/2K33+fubcggd0

const publicKey = `
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA3sQxyMwmDXQm3mumy345
E4NZID1guNsOYSMC7WDRbwpbCSnLDSXRZTiImdONjOTeH7UZtfF1G7YY78LrxYUS
x0dnqpAMfCgfJwt/JPMIJcfg5be2WaKMjC5uXK9Shlp9JUSFfV93YChcTkOTyv2j
CmQ+KU1P3UafpmKjCc5BwR9AIut6AGc6xtIfqb9VRa1hWGTS6XLuZQlH2Q0qYYnY
9zJVFPuBucV5t3w6brV8klpYs4OhRRKBn+2FI8bSltoreoTOdKpN5ot7mKK8uzaO
vjrHPUJbkwc0r6KUKTVqyBYz5GUAzt2ld6HYUP1g9zD6MGNOM/jEShThmDKSVm8G
DQIDAQAB
-----END PUBLIC KEY-----
`;


function verifySignature(data, signature, publicKey) {
  const verify = crypto.createVerify('RSA-SHA256');
  const parsedData = JSON.stringify(data);
  verify.update(parsedData);
  verify.end();
  return verify.verify(publicKey, signature, 'base64');
}

const result = verifySignature(data, signature, publicKey);
console.log('Is signature verified?', result);
```

Updated 5 months ago

← Webhook Events

Use Cases →