

Custom Webhooks

⚠ CAUTION

Webhooks are currently only available in certain [Hosted CTFd tiers](#) and [CTFd Enterprise](#).

Overview

Custom Webhooks are HTTP requests sent by CTFd to any endpoints you choose when certain [events](#) occur within the CTFd application.

CTFd webhooks go through an initial verification over `GET` requests. Once the webhook endpoint is verified, subsequent events are sent over `POST` requests.

Check out this tutorial on [how to integrate custom webhooks in your CTFd instance](#).

Endpoint ownership

Webhook endpoints are first validated by CTFd to prove that you have ownership over the endpoint.

Each CTFd instance that has the Webhooks plugin contains a randomly generated **Shared Secret**, which is used to prove the ownership of the endpoint.

Shared Secret

```
j371n6189gh618kat4h9y8j8g1ia7781718b8a9273728292giaj1h6a6y8j27jua
```

Validation

To validate an endpoint, CTFd will send a `GET` request to the target webhook URL with an additional query

parameter, `token`. The endpoint must then construct a JSON response containing the **token** and the **shared secret** hashed with **HMAC-SHA256**.

For example with an endpoint of `http://example.com/`, CTFd will send:

```
GET http://example.com/?token=2b00042f6
```

and your application should respond with

```
{"response": hmac_sha256(SHARED_SECRET, "2b00042f6")}
```

Events

Once the webhook endpoint is validated, you can then choose 4 event types that would trigger a `POST` request to the endpoint:

- User Events - a request is sent to the webhook endpoint when a **new user is created**
- Team Events - a request is sent to the webhook endpoint when a **new team is created**
- Solve Events - a request is sent to the webhook endpoint when a **challenge is solved**
- First Blood Events - a request is sent to the webhook endpoint when a **challenge is solved for the first time** and the solving account is not hidden or banned.

Event Examples

Event Header Example

Below is an example of the headers sent by CTFd when sending a webhook. Events types are specified by the `CTFd-Webhook-Event` header.

```
Host: example.com
User-Agent: CTFd-Webhook
Content-Length: 123
Accept: /*
Accept-Encoding: gzip, deflate
Content-Type: application/json
CTFd-Webhook-Event: team.created
```

```
Ctfd-Webhook-Signature: t=1616995588,v1=9e8cdd253434b74119ff44a119bb2d93e3b78a3a6a1a52d
Sentry-Trace: 3e8641d6696a4190ba634e35de7bcdtg-bv9f2938465208aa-1
X-Forwarded-For: 1.2.3.4
X-Forwarded-Proto: https
```

Request body

The **request body** contains the event body, particularly, in JSON format. The body will differ based on the event but in general will be the schema specified for the object in the [REST API documentation](#)

User Created Event

```
{
  "country": None,
  "website": None,
  "oauth_id": None,
  "name": "user",
  "id": 1,
  "fields": [],
  "team_id": None,
  "affiliation": None,
  "bracket": None
}
```

Team Created Event

```
{
  "country": None,
  "website": None,
  "oauth_id": None,
  "name": "Team Name",
  "id": 1,
  "fields": [],
  "captain_id": None,
  "members": [],
  "affiliation": None,
  "bracket": None
}
```

Solve Event

```
{  
    "challenge": None,  
    "team": None,  
    "date": "2022-09-27T06:21:53.210169+00:00",  
    "type": "correct",  
    "challenge_id": 1,  
    "id": 1,  
    "user": None  
}
```

First Blood Event

```
{  
    "challenge": None,  
    "team": None,  
    "date": "2022-09-27T05:55:38.106012+00:00",  
    "type": "correct",  
    "challenge_id": 1,  
    "id": 1,  
    "user": None  
}
```

Security

It's important to ensure that your endpoint verifies that the data is coming from CTFd. If not, malicious users could potentially replay requests to your endpoint or trick your endpoint into doing something it isn't supposed to.

CTFd provides the `CTFd-Webhook-Signature` header on every webhook sent out.

Take a look at this header taken from the example header [above](#):

```
CTFd-Webhook-Signature: t=1616995588,v1=9e8cdd253434b74119ff44a119bb2d93e3b78a3a6a1a52d
```

The header is split up into two sections `t` and `v1`.

`t` represents the unix time/epoch time when the webhook was sent out.

`v1` represents an **HMAC-SHA256** of the value of `t`, a **period (.)**, and the **request body**, with the **shared secret** as the HMAC key.

For example, if your endpoint received a body of `{'data': None, 'value': None}` then the signature would be made up of:

```
hmac_sha256(SHARED_SECRET, "1616995588.{'data': None, 'value': None}")
```

For optimal security, your endpoint should validate the signature of every request before using any of the data contained within. In addition, your endpoint can also choose to ignore requests whose signature is too old based on the timestamp.

Should an attacker attempt to manipulate the timestamp, request body, or signature, validation should fail as long as your shared secret hasn't been leaked.

Debugging

We provide simple debugging applications that implement the above security logic to help you debug/inspect webhook events. You can [check out this tutorial](#) on how to deploy them.

- Python
 - [app.py](#)
 - [Dockerfile](#)
 - [requirements.txt](#)
- PHP
 - [index.php](#)
 - [docker-compose.yml](#)

Questions/Feedback

Questions, feedback, and feature requests can be directed to <https://ctfd.io/contact/>.

Previous

[« Plugins](#)

Next

[Slack Webhooks »](#)

Was this page helpful?



[Share your feedback](#)

Docs

[Documentation](#)

Community

[MajorLeagueCyber](#) ↗

[Twitter](#) ↗

More

[Blog](#)

[GitHub](#) ↗

Copyright © 2025 CTFd LLC. Built with Docusaurus.