GitHub Token Inspection Tool

Caren Youssef

January 2026

Purpose

This document describes a small command-line tool that inspects a GitHub API token. The tool verifies whether the token can authenticate to GitHub, extracts basic account information, and reports the permissions (scopes) associated with the token. The output is designed to be easily consumed by other programs.

Approach

The tool performs a single authenticated request to GitHub's REST API endpoint `/user`. This endpoint is suitable because it returns the identity linked to the token when authentication succeeds.

The tool collects three types of information:

- **Authentication result**: success or failure, plus HTTP status code.
- **Identity signals**: selected user fields returned by the API (e.g., login, id, account type).
- **Token capability**: scopes read from response headers when available.

$$\begin{tikzcd}\text{GitHub API Token} \arrow[r] & \boxed{\text{Token Inspection Tool}} \arrow[r] & \begin{array}{c}\text{JSON Output}\\ \text{(valid / invalid,}\\ \text{user info, scopes)}\end{array}\end{tikzcd}$$

Implementation Summary

Inputs

The token can be provided either:

- via the command line option `--token`, or
- via the environment variable `GITHUB_TOKEN`.

The environment variable option is recommended to reduce accidental exposure in terminal history.

Request

The tool sends an HTTPS GET request to:

```
https://api.github.com/user
```

with the following headers:

- `Authorization: Bearer <token>`
- `Accept: application/vnd.github+json`
- `User-Agent: gh-token-inspect`

User Identification

When the request succeeds, the tool parses the JSON response body to extract basic information about the account associated with the token. This includes stable identifiers and public profile fields such as the login, user ID, account type, and creation date.

Scope Extraction

When present, the tool reads:

- `X-OAuth-Scopes` to obtain the token's scopes, and
- `X-Accepted-OAuth-Scopes` as an additional reference for the endpoint.

The values are normalized into a list by splitting on commas and trimming whitespace.

Operational Metadata

For observability, the tool also captures:

- response time (milliseconds), and
- rate-limit headers (`X-RateLimit-*`) when available.

Output Format

The tool always prints a single JSON object. This includes a boolean field `valid`, and either a `user` object (success) or an `error` object (failure). Request metadata is grouped under `meta`.

Example Output (Success)

```
{
  "valid": true,
  "user": {
    "login": "example_user",
    "id": 69876411,
    "type": "User",
    "name": null,
    "company": null,
    "blog": "",
    "location": null,
    "email": null,
    "public_repos": 3,
    "followers": 1,
    "created_at": "2020-08-20T14:38:28Z",
    "updated_at": "2026-01-09T18:35:14Z"
  },
  "meta": {
    "ok": true,
    "status": 200,
    "endpoint": "/user",
    "response_time_ms": 1818,
    "scopes": ["read:user", "repo"],
    "accepted_scopes_for_endpoint": [],
    "rate_limit": {
      "limit": "5000",
      "remaining": "4950",
      "reset": "1768119267",
      "resource": "core"
    }
  }
}
```

Example Output (Failure)

```
{
  "valid": false,
  "error": {
    "message": "Bad credentials",
    "status": 401
  },
  "meta": {
    "endpoint": "/user"
  }
}
```

Security Notes

- The tool does not print or store the token.
- Errors are handled defensively.
- It is recommended to use short-lived tokens and minimal permissions for testing.

How to Run

```
python github_token_inspector.py --token TOKEN_VALUE
```

or

```
$env:GITHUB_TOKEN="TOKEN_VALUE"
python github_token_inspector.py
```

Next Steps

- Some GitHub token types do not expose scope information through response headers, which limits direct permission visibility.
- Fine-grained tokens may provide less scope detail than classic tokens.
- The tool validates and inspects tokens but does not enumerate accessible repositories.
- A similar inspection tool could be implemented for GitLab API tokens.