


How To Hide API Keys, Credentials and Authentication Tokens on Github

 blog.netwrix.com/2022/11/14/how-to-hide-api-keys-github

Back in 2018, GitHub celebrated 100 million open source repositories, and it has only been growing since then. How can you make sure your sensitive credentials and authentication tokens aren't exposed to access by the public?

Handpicked related content:

[\[Free Guide\] Privileged Access Management Best Practices](#)

Read this blogpost to learn how to save your API keys and other important data from being disclosed.

How sensitive information gets exposed

Let's say I'm writing a script for an application that takes advantage of data from an API call. For example, this script targets weather data from OpenWeatherMap:

GET "https://api.openweathermap.org/data/2.5/weather?&id=5128581&appid={YOUR API KEY}"

It's not uncommon to store your API key (your secret) in a variable in the same file as the script, since this method makes testing quick and easy. Here's an example using Python and a fake API key:

```
# app.py
api_key = "12abc3d45ef6789012345g6789h0ij12"
city_id = "5128581"
base_url = "https://api.openweathermap.org/data/2.5/weather?"
final_url = base_url + "appid=" + api_key + "&id=" + city_id
```

Why this exposure is problematic

The problem is that when this code is pushed to a public GitHub repo, it's now exposing the secret API key to the world. This private access token should never be exposed outside of privileged users in your organization.

Some additional issues caused by this exposure:

- You may be in breach of your license agreement with the API vendor.
- Adversaries can use the stolen key to make rapid API calls outside the scope of your license, causing the API vendor to throttle your requests.
- Most important, if the private API key gives access to sensitive data such as a cloud storage account, then you're essentially inviting a security breach.

How sensitive data can be secured

So how can we hide API keys in Python or another language on Github? Well, the solution is simple. We just need to configure a file that stores our API keys (and other sensitive data) and that is included in other code files when necessary but ignored by version control (e.g., gitignore). This allows your application to function as expected while preventing any sensitive credentials from being pushed to GitHub.

Revisiting our earlier example, we can create a second code file named config.py to store any code needed to access our API key. Then all we need to do to make it work is to change the strings in app.py as shown in red to get the API key from config.py rather than assigning it to a variable directly:

```
# config.py
api_key = "12abc3d45ef6789012345g6789h0ij12"

# app.py
import config
city_id = "5128581"
base_url = "https://api.openweathermap.org/data/2.5/weather?"
final_url = base_url + "appid=" + config.api_key + "&id=" + city_id
```

Now we need to prevent config.py from being pushed to GitHub by adding it to our gitignore file:

```
# .gitignore
config.py
```

When we push to GitHub, only app.py and .gitignore will be uploaded to the public repository. Our config.py file that contains all the sensitive information will not end up on GitHub.

Removing sensitive data you've already pushed to GitHub

If any of your sensitive data has been pushed to GitHub in the past, follow [GitHub's guide](#) for removing that data from a repository.

Conclusion

This is a simple solution, but a powerful best practice any time you're dealing with credentials. In fact, I never assign secure credentials to variables unless the file is excluded from version control.

Depending on your organization's standards, you may want to also apply this to self-hosted version control. In general, it's a good idea to know exactly where sensitive credentials are stored. and pushing them to version control is often not a secure practice regardless of where version control is hosted (local, cloud, etc.).

FAQ

Do I need to hide my API keys?

API secret keys should never be put in a client-side code or should be hidden. However, read-only API keys won't pose any risk if you paste them into your JavaScript code that will commit in your browser.

Should API keys be stored in git repositories?

Putting API keys in public or private git repositories puts them at serious risk of being exposed. If you choose to do so, be sure to encrypt your sensitive data with [git-remote-gcrypt](#), [git-secret](#) or [git-crypt](#).

How does dotenv help with hiding API keys?

[Dotenv](#) is a module that loads environment variables from a .env file into [process.env](#). Basically, it helps to make your API key invisible in the code.

Joe Dibley

Security Researcher at Netwrix and member of the Netwrix Security Research Team. Joe is an expert in Active Directory, Windows, and a wide variety of enterprise software platforms and technologies, Joe researches new security risks, complex attack techniques, and associated mitigations and detections.

