rOpenSci - open tools for open science

A better way to manage your Github personal access tokens

Jeroen Ooms

Published July 7, 2020

Citation

Ooms, J. (2020, July 7). A better way to manage your Github personal access tokens. *rOpenSci - Open Tools for Open Science*. https://doi.org/10.59350/ezs92-3kz34

Keywords

Package, Gert, Credentials, Git, Github

We have been working hard behind the scenes on the upcoming release of our new git package named gert, a joint effort from rOpenSci and the Tidyverse team. One of the main features of gert is the out-of-the-box authentication mechanism, which is provided via the new credentials package.

Among other things, the credentials package makes it possible to save and load https authentication details from the git credential store, which is part of the official command-line git. Thereby credentials are automatically shared between command line git and the gert package, while safely stored by your operating system's preferred password manager.

In this post we show how you can take this one step further, and use the credentials package to save your `GITHUB_PAT` in the git credential store. This way you can authenticate with the GitHub API using the same token that is used for HTTPS remotes in git and gert. This is convenient for users, and also provides package authors with a mechanism to prompt the user for credentials, without having to take responsibility for managing tokens.

Note for Windows users: the credentials package requires a recent version of Git for Windows.

What is a Personal Access Token

GitHub allows you to generate Personal Access Tokens, which you can use instead of your password when authenticating over HTTPS, both for git remotes and the GitHub API. Major advantages are:

- If you have enabled two-factor authentication (2FA), you must use a PAT to authenticate programmatically. You have no choice.
- You can generate many PATs with specific permissions, giving you fine-grained security control.
- A PAT can easily be revoked or replaced with a new one.

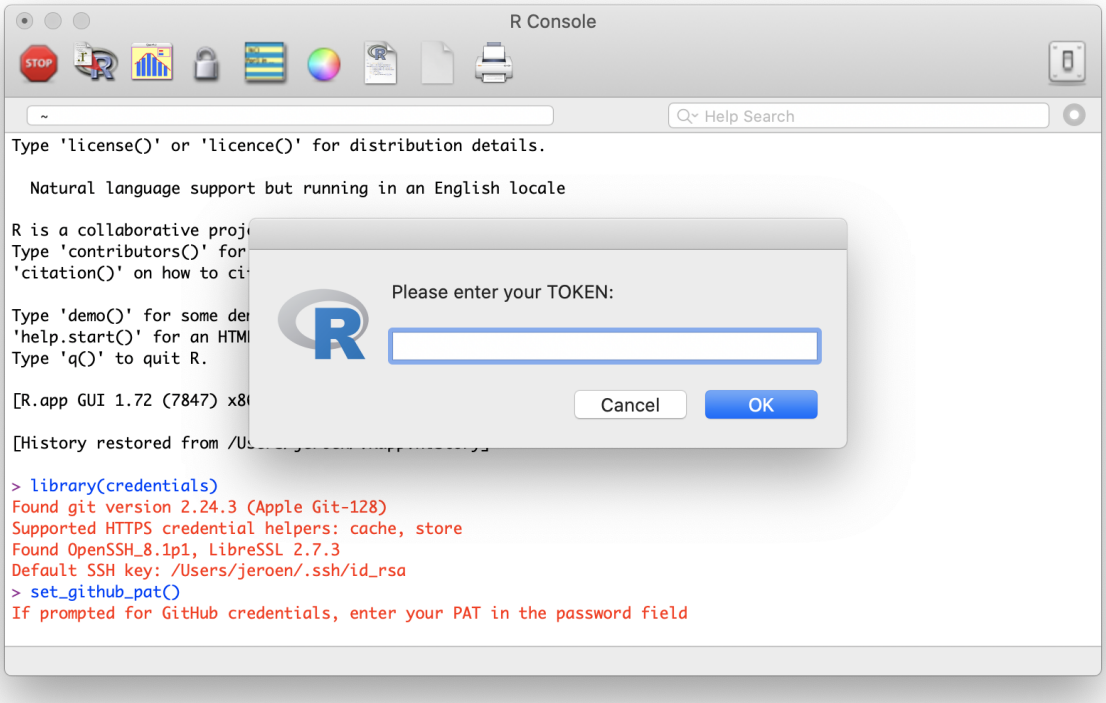
In conclusion, if you are a responsible GitHub user, you have enabled 2FA on your account, and you only ever enter your main password when authenticating on the GitHub website. Everywhere else you should be using a PAT, preferably one that only has the permissions it needs.

Storing and loading your GitHub PAT

Most R packages that interact with the GitHub API expect that your PAT is stored in the environment variable `GITHUB_PAT` in the R process. But how does it get there? The credentials package provides a function that will set this environment variable:

```
# Tries to set the GITHUB_PAT environment
variablecredentials::set_github_pat()## TRUE
```

This function calls out to the git credential store to get a suitable token for the `github.com` domain. If no token is available yet, the git credential manager will then prompt the user to enter one. What this looks like depends on your operating system and credential helper configuration.



The `set_github_pat()` function returns `TRUE` when it succeeds in setting the `GITHUB_PAT` environment variable, and `FALSE` if not. Packages that call `set_github_pat()` to let the user authenticate, can check the return value to determine if authentication was successful.

Once a working PAT has been stored in the git credential store, it can automatically be loaded in another R session by calling `set_github_pat()` again. The token is automatically validated, and if it still works, the `GITHUB_PAT` environment variable is set without the user having to do anything.

And here is the best part: because the token is stored under the `github.com` domain in the credential store, both gert and command line git will automatically attempt to authenticate with this token when fetching/pushing Github HTTPS remotes.

Managing the credential store

How long does the credential store remember your token? This depends on which credential helper is in use. On all systems, git includes at least the following helpers:

- `cache`: *Cache credentials in memory for a short period of time. See git-credential-cache for details.*

- `store`: *Store credentials indefinitely on disk. See git-credential-store for detail*

However on MacOS and Windows, git actually defaults to custom credential helpers that use the OS password manager. On MacOS this is called keychain and on Windows this is git credential manager for Windows. These credentials helpers can store your credentials indefinitely.

```
credentials::credential_helper_get()## [1] "osxkeychain"
```

The credentials package includes a few more utility functions to help you interact with the credential store. To manually load credentials for a given domain use `git_credential_ask`:

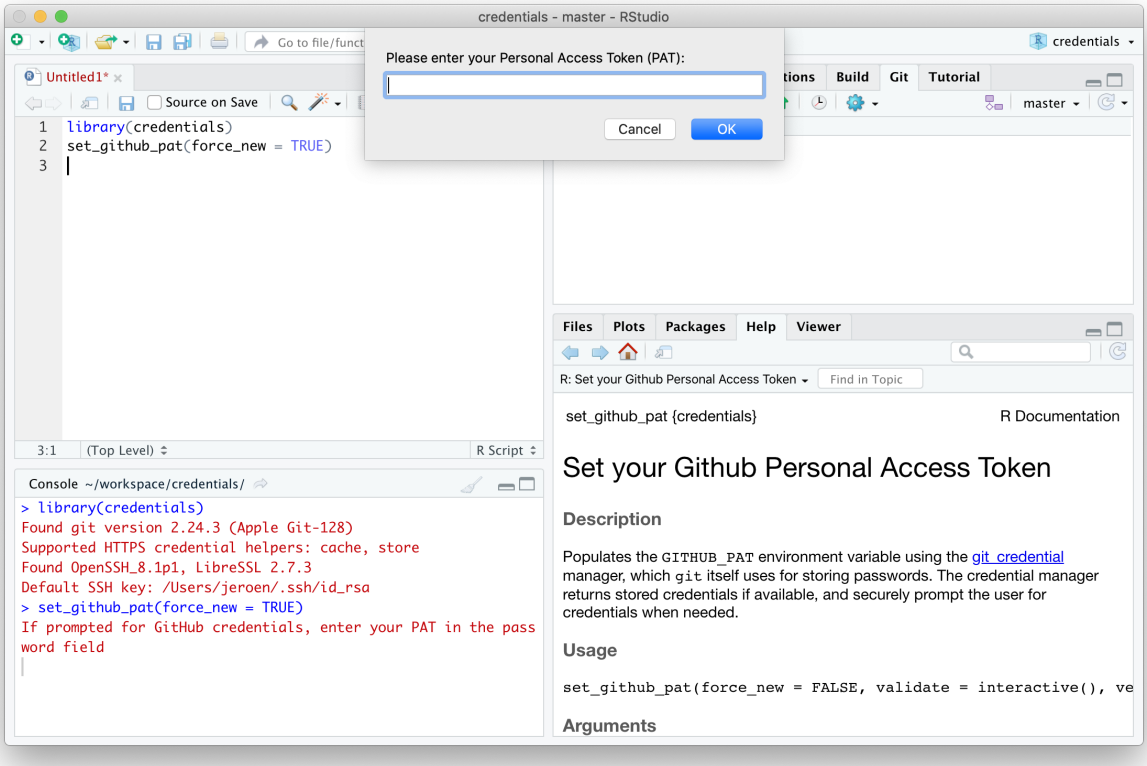
```
credentials::git_credential_ask("https://github.com")## $protocol## [1]
"https"## ## $host## [1] "github.com"## ## $username## [1] "token"## ##
$password## [1] "aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d"
```

Use `git_credential_forget` to explicitly remove a credential:

```
# Drop GITHUB_PAT credential from
storecredentials::git_credential_forget('https://token@github.com')
```

Alternatively, if you want to switch to another PAT, use `set_github_pat(force_new = TRUE)`. This will automatically drop any existing PAT from the credential store, and always prompt the user to enter a new one.

```
# Drop existing GITHUB_PAT and ask for a new
onecredentials::set_github_pat(force_new = TRUE)
```



Workflow

We still need to figure out how this will affect the recommended workflow. Currently many users hardcode the GITHUB_PAT in the `~/.Renviron` file, so that it is automatically set in every R session. You could accomplish the same with the credentials package by adding this to your `~/.Rprofile` file:

```
# Load the GITHUB_PAT in the R sessioncredentials::set_github_pat()
```

However perhaps it is actually undesired to always have your GITHUB_PAT exposed in R. The nice thing about the credentials package is that it becomes easy to load your access token *on demand*. Hence, instead of setting the PAT on the start of each R session, a user or 3rd party package could call `set_github_pat()` whenever it needs access to the Github API.