Intermediate Git and Github: Security

An Introduction

Jenny Leopoldina Smith

Fred Hutchinson Cancer Research Center

2021-09-27

Introduction

- I will be discussing some a very **general overview** of security when using Git and Github

- This is not an exhaustive list
  - The goal is only to spread awareness of security issues
  - *Security practices change over time*
  - Be sure to periodically check around Github's blogs, or software you use (an example would be Docker containers or cloud services), or a friendly co-worker, occasionally to discuss

Protect Against Secrets in Git Repositories

- This is an overview of the "best practices" highlighted in a post by the computational biologist Sean Davis[1,2], but its relevant to everyone.
- In this example, the author using github included their secret key for AWS directly into code + committed the change in a local repo.
    - The key was used to access AWS and use resources the author was paying for!
- To be proactive - check out `git-secrets` to secure your passwords and sensitive information!

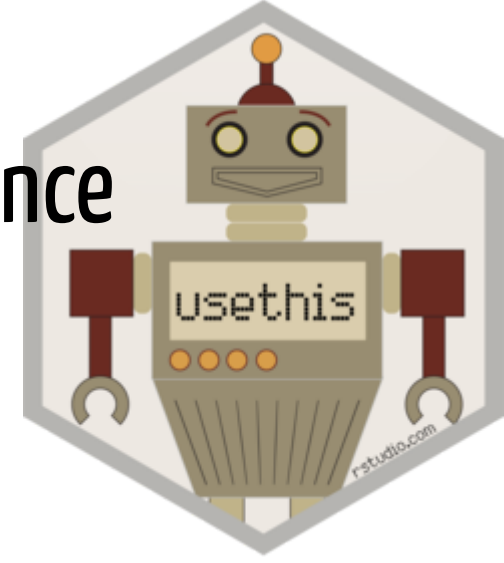
[1] Sean Davis, post on Medium

[2] Sean Davis, post on Personal Blog

[3] Image credit: github blog

git-secrets

- The code snippet is directly from the blog post.
    - If you use a Mac, installation is easy with `homebrew`
- To try it out, you can make toy git repo to test the functionality.

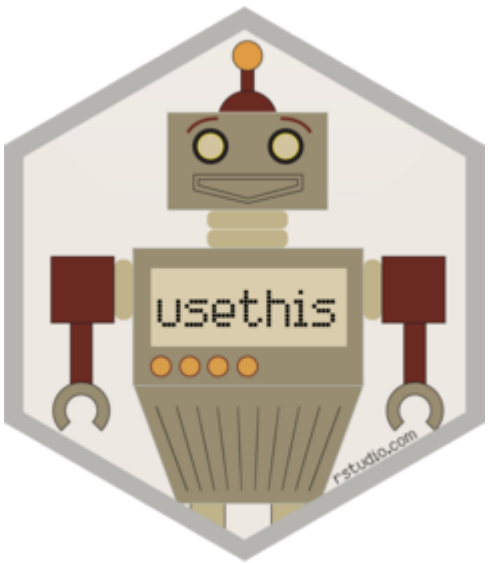
```
```{bash}
# create an example git repo
mkdir secrets_example  #<<
cd secrets_example  #<<
git init  #<<
# now "install" the git-secrets hook
git secrets --install  #<<
```
```

git-secrets

- once installed, list any secrects detected in your environment with `git secrets --list`
- Then add a simple example password to git-secrets configuration
    - Try to commit a file containing your password

```
```{bash}
git secrets --add 'MyPASSWORD[0-9]+' #<<
echo 'MyPASSWORD123' >> test.file
git add test.file #<<
git commit -m 'test file with password' #<<
```
```

- the output: `[ERROR] Matched one or more prohibited patterns`

Additional Info: best practices for AWS keys

R package: `usethis` for Guidance



- The package `usethis` provides a fantastic overview of getting started with git and github for beginner to advanced users.

- The vignette covers the following concepts:
  - How to get a free GitHub.com account and install Git.
  - Configure your Git user.name and user.email
  - Reviews how RStudio can find your Git executable
  - Reviews how you can pull/push from your local computer to GitHub.com, in general and from RStudio
  - Reviews how to get a personal access token (PAT) from GitHub.com and make it available in R sessions

R package: `usethis` for Guidance

- To begin, install the package and configure your .Rprofile and git.

```
```{r tidy=TRUE}
library(usethis)
usethis::edit_r_profile() #Set HTTPS for github protocol
use_git_config(user.name = "Jane Doe",
               user.email = "jane@example.com")
```
```

- Some useful functions from `usethis` for use with github are:
  1. `create_project()`
  2. `use_git()` and `use_github()`
  3. `git_vaccinate()`
  4. `git_sitrep()`
  5. `gh_token_help()`

> `git_vaccinate()` Adds .DS_Store, .Rproj.user, .Rdata, .Rhistory, and .httr-oauth to your global (a.k.a. user-level) .gitignore. This is good practice as it decreases the chance that you will accidentally leak credentials to GitHub.

R package: `usethis` for Guidance

- To enable git and github, you will have to configure your PAT (personal access token)
    - It might seem overwhelming, but getting and setting up your PAT can be done!
- Again, there is a full article from `usethis` covering the topic: "Managing Git(Hub) Credentials".



Setting up Git Credentials

1) MacOS and Linux to store credentials for GitHub over HTTPS

- Installatipn of a credenial manager `brew install --cask git-credential-manager-core`
- "The next time you clone an HTTPS URL that requires authentication, Git will prompt you to log in using a browser window."

2) Another useful follow-up function is `usethis::gh_token_help()` which can provide helpful prompts!

```
• GitHub host: 'https://github.com'
• Host online: TRUE
• Personal access token for 'https://github.com': '<discovered>'
ℹ Call `gh::gh_whoami()` to see info about your token, e.g. the assoc
ℹ To see or update the token, call `gitcreds::gitcreds_set()`
✓ If those results are OK, you are good go to!
ℹ Read more in the 'Managing Git(Hub) Credentials' article:
  https://usethis.r-lib.org/articles/articles/git-credentials.html
```

Mistakes Happen

- Remember security is a community issue, and its always evolving.
- For example, the vignette from `usethis` has changed significantly over the last few years from storing your `.Renviron` file.

> This means we can stop storing GitHub PATs in plain text in a startup file, like .Renviron 4. This, in turn, reduces the risk of accidentally leaking your credentials.

- Leaks can and will happen.
  - And there are active pushes to address these, with secret scanning enabled in github repos
  - As recently as mid-September 2021,`travis-ci` leaked nearly all users keys, passwords, and other sensitive information.

Thanks!

Please feel free to contact me: *JennyL.Smith12 [at] gmail.com*

[1] Slides created with the R package **xaringan**.