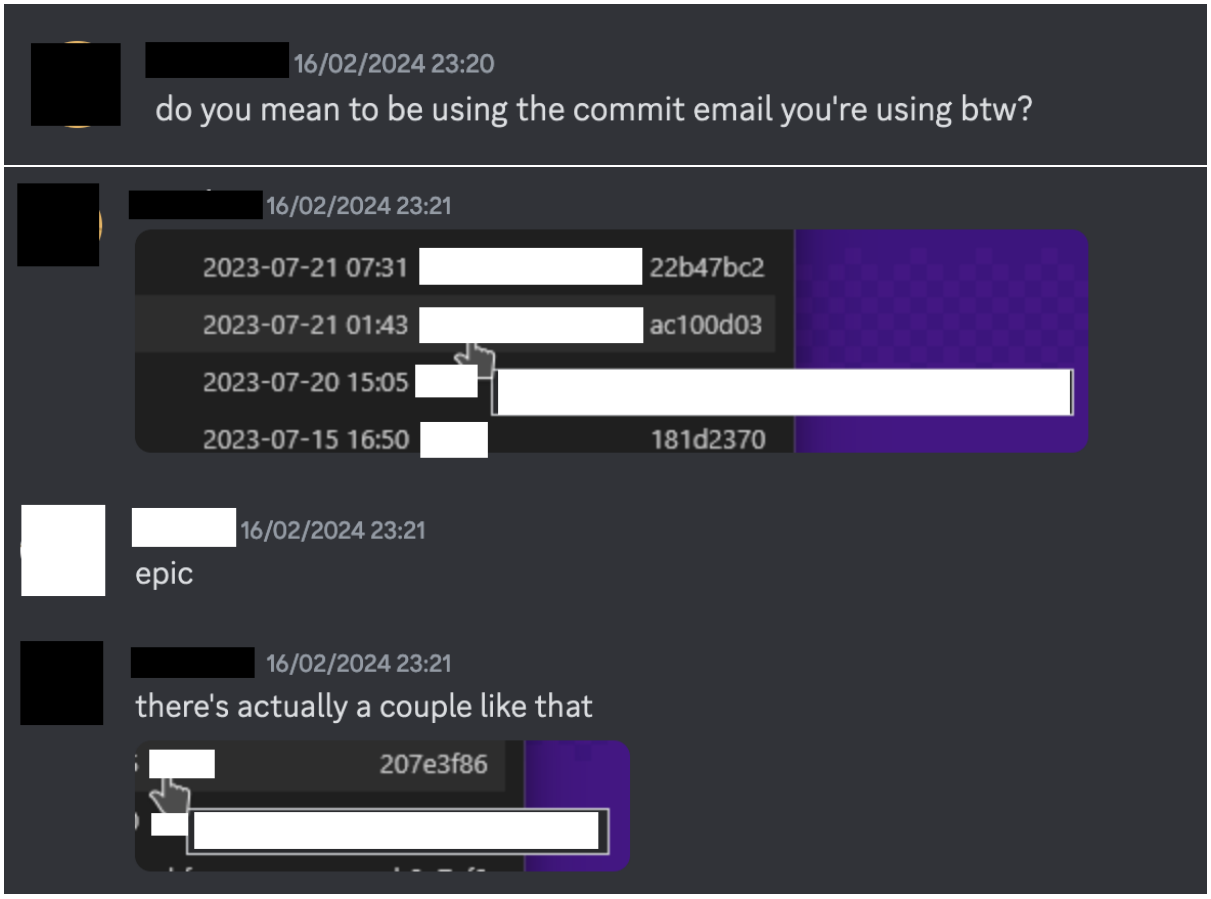
How to doxx yourself using git and GitHub (and how not to)

In this short article, I'll discuss an experience I've had in which I accidentally exposed my personal e-mail address in my GitHub repositories. And how I used special tools to fix it all at once.

How it happened

A few months ago, I was updating my personal blog. It is a simple static website made using HUGO, and generously hosted by a friend of mine. I also decided to make the source code of the blog public and link it from the front page.

As I published it, my friend sent me a peculiar message:



There they were... full name and email address right in the commit history. In a display of tech illiteracy, back when first configuring git, I simply entered my actual email, along with my full name, when prompted. And all this time, I've been committing with this configuration.

While largely inconsequential in my case, it still feels disagreeable to share too much online. People may want to keep their "work" and "not-work" internet identities separate, and my commits have been happily mixing them together.

Time to fix the issue then.

GitHub's noreply email

Now aware of the issue at hand, I got to searching. I open Firefox, and do a classic keyword search: "github keep email private"

Well, that was easy.

In short: GitHub provides you with a fake email address, called "noreply address", that only exposes your GitHub username.

You can find the address in your settings and can also configure your GitHub account to reject any commit that does NOT come from your noreply address.

Batch modifying existing commits

Now all I had to do was to change the commit email for every commit in my blog's repository.

So how do you do that exactly?

...

...

Firefox > "git change email for all commits"

And just like that, I knew how!

After cloning the repository to a new folder (for safety's sake since we will be overwriting a whole git repository), it's as easy as:

~~Setting up an alias in your git config to handle the batch operation!~~ Wait that sounds tedious, I'd rather not define macros when possible...

~~Using the git-filter-branch tool!~~ Um... This one is apparently very slow and a bit dangerous, let's look a bit further...

Using the git-filter-repo tool! And that's why, when it's about something I'm not familiar with, I always explore past the first answer in Stack Overflow, even if it's the accepted answer.

Anyways, all I need to do now is install and run it:

```
sudo apt install git-filter-repo
git-filter-repo --name-callback 'return name.replace(b"MyActualName", b"MyUserName")' --email-callback 'return email.replace(b"myactual@mail.com", b"mynoreply@mail.com")'
git push --force
```

And with that, I was able to replace all occurrences of personal information in my git Commits!

Careful

Remember that doing batch operations on an entire repository carries risk, please do those manipulations on a separate clone of the repository to keep a backup in case something goes wrong.

Conclusion

In this short article, I showed what happened if you naively enter correct information when configuring git, and how to resolve the issue on an existing repository using third party tooling, with a small tangent on proper Googling etiquette.

Thanks for reading. See you in the next post!