# Submitting New CVEs Using GitHub - Initial Tools Setup

*Author: Madison Oliver; July 2019*

The purpose of this document is to discuss the several steps in setting up the correct environment to submit a CVE through GitHub using a variety of tools. This assumes that you already have approval and recognition from MITRE that your organization is participating in the GitHub pilot program, and that you have git installed - <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>. The GitHub initial setup should be completed regardless of the submission process. The SourceTree setup is optional, and should only be completed if you intend to use SourceTree for submission.

[GitHub Initial Setup](#_gjdgxs)

[CLI Setup](#_30j0zll)

[GitHub Desktop GUI Setup](#_4d34og8)

[SourceTree Initial Setup](#_3rdcrjn)

# GitHub Initial Setup

1. Start with an individual GitHub (GH) account. Use one that you have (with your organization’s approval) or make a new one - <https://github.com/join>
	1. If you already have a GPG key that you plan on signing your commits with, now is a great time to add it to your GH account (assuming its with the same email address that is used with your GH account)
		1. How to add a GPG key to your GitHub account - <https://help.github.com/en/articles/adding-a-new-gpg-key-to-your-github-account>
	2. If you don’t have a key, making one is easy! - <https://help.github.com/en/articles/generating-a-new-gpg-key>
		1. *When asked to enter your email address, ensure that you enter the verified email address for your GitHub account. To keep your email address private, use your GitHub-provided no-reply email address.*
	3. More information about signing commits - <https://git-scm.com/book/en/v2/Git-Tools-Signing-Your-Work>
2. Your organization will need to have a GH account. If you are not sure if your organization has a GH account, please search GH or ask internally. If they do have one - great! If not, make one (with approval).
	1. Here’s an example of CERT/CC’s GitHub <https://github.com/CERTCC>
	2. If your organization already has a GitHub organization account and it is managed elsewhere/not by your team, you should have them **add** your GitHub account to the organization - <https://help.github.com/en/articles/organizing-members-into-teams>. They will need your GitHub account username (and possibly more information) to do this. Additionally, if another team controls the organization’s GitHub, they will likely need to do Step 3, depending on your permissions.
3. Create a fork of MITRE’s cvelist GH repository within your organization GH
	1. A [fork](https://guides.github.com/activities/forking/) is a special kind of copy of MITRE’s repository that allows you to contribute back to it (like adding cve files!). You will have a copy of this within your organization’s GH. This is where you will internally push your CVE .json files. Keep in mind - once these files are on this GH fork, *they are essentially public information as anyone can see this!*
	2. MITRE’s repository - <https://github.com/CVEProject/cvelist>
	3. Here’s an example of CERT/CC’s fork of MITRE’s repo - <https://github.com/CERTCC/cvelist>
	4. Steps on how to fork a repo in GitHub - <https://help.github.com/en/articles/fork-a-repo>

#

# CLI Setup

1. Create a local clone of your organization’s GH fork of MITRE’s repo on your host machine
	1. **$ git clone** [**https://github.com/YOUR-USERNAME/YOUR-REPOSITORY**](https://github.com/YOUR-USERNAME/YOUR-REPOSITORY)
2. Enable GPG signing for commits
	1. Find your key ID by using
		1. **$ gpg --list-keys**
	2. Enable signing globally
		1. **$ git config --global user.signingkey KEY-ID-HERE**
	3. Don't have a key? Make one!
		1. **$ gpg --gen-key**
3. Add your organization’s GitHub repository as a remote repository to your local clone. This will allow you to push your changes and new .json files to your repository on GitHub to then get it to MITRE.
	1. Sets the new remote
		1. **$ git remote add origin** [**https://github.com/YOUR-USERNAME/YOUR-REPOSITORY**](https://github.com/YOUR-USERNAME/YOUR-REPOSITORY)
	2. Verify the new remote URL → if this command doesn’t generate a response then there isn’t a remote repo setup
		1. **$ git remote -v**

#

# GitHub Desktop GUI Setup

<https://desktop.github.com/>

GitHub already has great documentation on this - please follow their steps here to install and setup the GUI <https://help.github.com/en/desktop/getting-started-with-github-desktop>

1. In addition to the steps above, you should create a local clone of your organization’s fork of MITRE’s GitHub
	1. In the GH Desktop client, File → Clone Repository. If you’ve already logged in with your GitHub account, the available repositories should automatically show up. Choose your organizations CVE list fork and click “Clone”.
		1. 
		2. You could additionally clone a repository by pasting the appropriate URL.
			1. 
2. GitHub Desktop GUI does not allow for signing of commits on the GUI, but since it essentially runs git commands in the background, you can setup the git configuration file to automatically sign all commits by default. (Signing commits is a good idea anyway so you may as well set that up now.)
	1. To configure your Git client to sign commits by default for a local repository, in Git versions 2.0.0 and above, run
		1. $ git config commit.gpgsign true
	2. To sign all commits by default in any local repository on your computer, run
		1. $ git config --global commit.gpgsign true

#

# SourceTree Initial Setup

1. Install SourceTree (ST) if you prefer over GitHub Desktop or git cli: <https://www.sourcetreeapp.com/>
	1. This will require an Atlassian account - make one if you don’t have one
	2. Sign into ST using your GH account when prompted
		1. 
2. Within ST, create a local clone of your organizations GH fork of MITRE’s repo
	1. Example, CERT/CC’s fork - <https://github.com/CERTCC/cvelist>
	2. 
	3. This will create a local clone (basically a copy) on your local machine at the Destination Path specified
3. Set up ST to enable GPG signing for commits with your GPG key
	1. Menu Bar → Repository Settings → Security, check the box that says “Enable GPG key signing for commits”