# Submitting New CVEs using SourceTree

*Author: Madison Oliver; July 2019*

The purpose of this document is to describe the GitHub submission process using SourceTree, free GUI-based software. This can also be replicated with other GUIs → <https://git-scm.com/downloads/guis>. The following steps assume that your git environment is already set up (if it isn’t, please review the Setup document).

[Submission](#_gjdgxs)

[Updating](#_30j0zll)

# Submission

1. In SourceTree (ST), while working in your local clone of your fork of your organizations GH:
	1. Create a branch to work in named "cna/[ORG]/[SOME IDENTIFIER]", where "SOME IDENTIFIER" could be:
		1. the CVE ID if it is a single CVE
		2. The organization’s publication tracking number if it is multiple CVEs from one single vulnerability publication
		3. A description of its purpose if it is from multiple publications
		4. The date of the submission in YYYYMMDD format
		5. 
		6. Checkout the new branch by double clicking it under “Branches” on the side bar if you did not have the "Checkout new branch" checkbox checked when creating the branch (this is checked by default)
2. Outside of ST, copy your CVE JSON file(s) to the appropriate directory within the local clone of your fork of the organization’s GH
	1. Copy (or move) your JSON files to the appropriate directory - they are organized by year and then by leading numbers, e.g. CVE-2018-1234 would go in the 1xxx folder within the 2018 folder
	2. Replace the existing files (they only contain placeholder text) with your new JSON files
	3. 
	4. File Status in the top-left of ST should show that there are uncommitted changes since your new files have been added to the appropriate directory but a commit has not been created yet
		1. 
	5. Commit the changes by clicking "Commit" in the top left while on the "File status" tab
		1. Check the box next to "Unstaged Files" to move them to "Staged Files"
		2. Add something to the commit message to show what this commit contains
		3. Under "Commit Options", be sure that "Sign commit" is selected
		4. Click "Commit" in the bottom right to commit it
		5. 
			1. The staging environment is where git stores files that will be included in the next commit ("git add" on the command line). In the way you’ll be using it, all the files can likely go into the staging environment and into one single commit.
			2. If someone had multiple vulnerability reports with multiple CVEs, they could add the CVEs from report #1 to the staging environment, commit it, and then add the CVEs for report #2 to the staging environment, and commit it. This would generate two commits that contain the CVE specific to each vul report. The benefit of this would be clarity as the commit messages are seen publicly, but isn’t necessary.
	6. Push this new commit to GH by selecting "Push" in the top bar
		1. Select the branch that you want to push (cna/certcc/CVE-2018-5678 in this example) and then click "Push"
		2. 
		3. If this is successful, you will receive a "Completed successfully" message. If you do not receive that or an error message, it is likely that you did not choose which branch to push.
		4. 
3. On GitHub (<https://github.com>), you should be able to initiate a pull request to MITRE’s repository
	1. There should be a green box that says "Compare and pull request" that you can click to generate the pull request
		1. 
			1. If that box is not there, just click "New pull request". The box is generated by GitHub acknowledging differences between MITRE's CVE list and your forked copy. Sometimes it isn't immediate recognize it depending on how quickly you do this, so if that box is not there it doesn't necessarily mean the changes aren't there.
	2. Initial the pull request, put an appropriate message in there (it can be the same as the commit message) and submit it to MITRE
		1. They will come back with any requested changes if needed. To make these changes, update the files on your host machine in the same repo and branch that you initially put it in, commit the new changes, and then push them to Github. They'll automatically show up as part of this pull request as a new commit.