Nuclei

Nuclei is a fast and efficient open-source tool that helps with scanning vulnerability classes, SQL injection, and remote command execution. Nuclei can identify potential vulnerabilities in the software, making it useful in developing software while ensuring security and stability. Nuclei can be integrated with the GitHub CI/CD pipeline so whenever an issue is caught then the developers are reminded about it.

How was it implemented?

To implement Nuclei, a yml file was created inside the .github/workflows folder and it will be triggered whenever a push occurs. It uses a vulnerability scanner library called **projectdiscovery/nuclei-action@main** to target our website **http://localhost:3000/**.

How did it help?

Nuclei helped in finding potential vulnerability inside the project by scanning the files every time a push was triggered to ensure security and stability was handled properly throughout the project.

Figure 1: Diagram of CI/CD for Nuclei

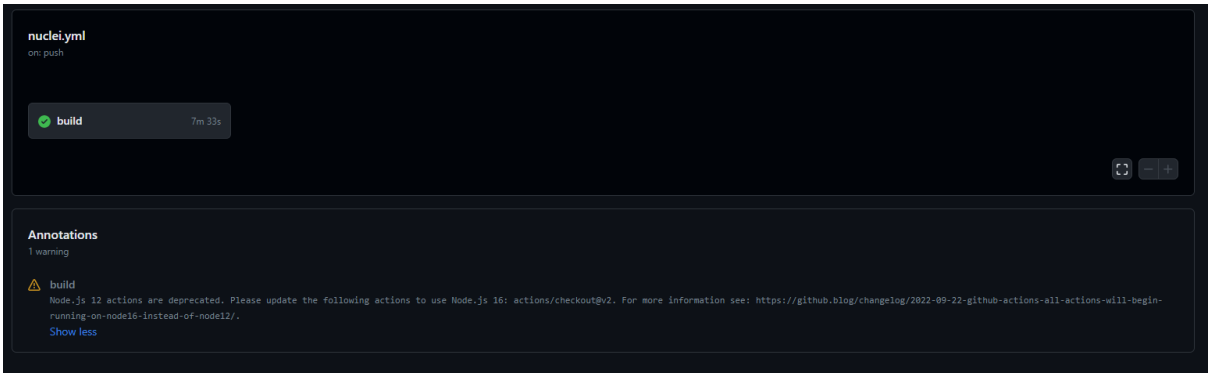

Figure 2: Illustration of Nuclei detecting a potential vulnerability

Semantic Commits

The checker for semantic commit using CI/CD ensures following:

1. Since commits are structured, contribution to the projects becomes easier.
2. It is also easier to understand the entire change made by that commit since semantic commit makes it informative.
3. The process of creating a release will become easier.

How was it implemented?

To implement semantic commit, a yml file was created inside the .github/workflows folder and it will be triggered whenever a pull request was created. It uses a semantic commit library called **amannn/action-semantic-pull-request@v5** which ensures that all commits inside a pull request are semantic commits.

How did it help?

Semantic commits help in understanding the change that was made with a specific pull request just by reading an informative semantic commit. Semantic commits have following semantics:

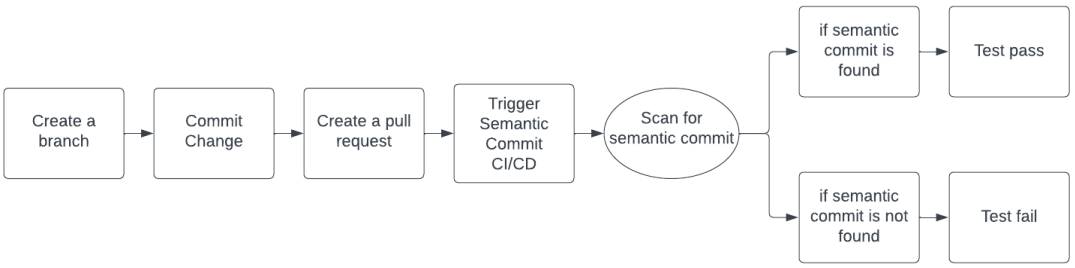

Figure 3: Diagram of CI/CD for Semantic Commit

feat: (new feature for the user, not a new feature for build script)
fix: (bug fix for the user, not a fix to a build script)
docs: (changes to the documentation)
style: (formatting, missing semicolons, etc; no production code change)
refactor: (refactoring production code, eg. renaming a variable)
test: (adding missing tests, refactoring tests; no production code change)
chore: (updating grunt tasks etc; no production code change)

Figure 4: An illustration of semantic commit being checked

Why did we choose to integrate Nuclei and Semantic Commit for CI/CD compared to other tools?

The primary reason for implementing nuclei and semantic commits as our CI/CDs was to ensure that as the project scales further, these CI/CDs will make the debugging easier while ensuring security at all times.

References

1. H. Bothra, “Implementing nuclei into your github CI/CD pipelines,” *ProjectDiscovery.io*, 25-Jan-2023. [Online]. Available: https://blog.projectdiscovery.io/implementing-nuclei-into-your-github-ci-cd-for-scanning-live-web-applications/. [Accessed: 29-Mar-2023].
2. “Semantic-pull-request - github marketplace,” *GitHub*. [Online]. Available: https://github.com/marketplace/actions/semantic-pull-request. [Accessed: 29-Mar-2023].
3. “Semantic commit messages,” *Gist*. [Online]. Available: https://gist.github.com/joshbuchea/6f47e86d2510bce28f8e7f42ae84c716. [Accessed: 29-Mar-2023].