



CVE Binary Tool

Improve language-specific package support

About Me:

Name: Anant Vijay

GitHub: [XDRAGON2002](#)

Email: anantvijay3@gmail.com

University: Indian Institute of Information Technology, Gwalior

Program: Bachelor of Technology in Computer Science and Engineering

Year: 2nd year

Location: India

Timezone: Indian Standard Time (GMT +5:30)

Portfolio: [DRAGON2002](#)

The GSoC timeline is mostly in sync with my university's summer break and thus will allow me ample time to work on the project. Even after the break ends there would be no tests or examinations as well as the classes would be asynchronous leaving me enough time to work. I have been contributing to the project since around last year end and hence have a good idea regarding the existing project structure which would be extremely helpful while working on the project. Moreover I have a lot of experience with working on python which again would come in handy. I am really excited to spend my summer working on this project!

Code Contributions:

[Pull Requests](#)

[Issues](#)

I have gone through the entire codebase a good few times and believe that I have understood the functioning of the tool and hence would be able to work on the project comfortably without facing a lot of issues or difficulties.

Project Abstract:

- Create/improve structure of package data parsers:
 - ◆ Create an OOPs based structure to add parsers
 - ◆ Restructure existing parsers to the new format
 - ◆ Wrap the parsers under a uniform abstracted API for parsing package data


- Add support for new languages and package managers and improve existing ones

Detailed Description:

Create/improve structure of package data parsers:

Currently the existing structure for the package data parsers is very haphazard and scattered in nature, on further inspection it is obvious that various commonalities emerge amongst them. They all take a specific type of file to parse, extract the dependencies from it and query the NVD database for vulnerabilities regarding those dependencies.

These parsers follow a similar program flow, similar input/output formats and are also independent of each other, hence, act as black boxes. I propose to use Object Oriented



Programming Paradigm in order to properly structure these parsers on the basis of class method overriding.

The proposed structure would look very similar to the current structure for checkers in which there is a base parent class that every checker inherits from and is modified accordingly for different packages.

Two basic functionalities of every parser is to extract the product and version pairs from the package file and then extract the vendors to encompass the Product Info for that specific dependency, these functionalities would act as the core member methods.

In cases where an extra steps, i.e. use of subsystems to extract the list of installed packages or some other step specific to the parser in question, another class method could be implemented to do the same, maintaining the modularity of the code for those special cases and would act as internal helper methods.

This structure would reduce the complexity of the code, improve modularity and abstraction further and make it easier for others to add new package data parsers.

I also propose that after this structure is implemented the currently present package parsers would also be refactored to conform to the same system.

Once all the present parsers are refactored, an API wrapper could be employed over them which would simply take in a file format and internally handle the parsing, hence further improving upon the layer of abstraction and providing convenience for the users.

Add support for new languages and package managers and improve existing ones:

As the tool is trying to improve its support to various new formats which were not supported earlier, I propose to incorporate support for popular languages and package managers and improve upon the language support that are currently present. The list includes working on but is not limited to:

- Python
- Javascript
- Rust
- R
- Golang
- Ruby

If time permits, support for more languages/package managers will be added. I would also like to improve the current package list parsers as an extended goal. I also have a fair bit of experience working with most if not all of these above-mentioned languages and am familiar with their package management process as well which would be helpful in parsing the language package lists.

Moreover I plan to add support for the new languages after the refactor for the existing parsers is done because of which the process of addition of new parsers would be thoroughly tested as well as the parser API would also be properly iterated over.

Weekly Timeline:

→ Pre GSoC:

- ◆ Contribute to the project by working on issues to further improve my understanding of the codebase.
- ◆ Look at various languages and select the ones that I would be adding support for with proper discussion with mentors.

→ Community Bonding Period (May 20 - June 12):

- ◆ Bonding with the community actively.
- ◆ Discussing and refining the project idea with the help of the community and the mentors.
- ◆ Create a list/collection of issues that I would be working on.

→ Week 1 (June 13 - June 20):

- ◆ Look into the commonalities between the language parsers in detail.
- ◆ Create a basic class structure based on the commonalities.

→ Week 2 (June 20 - June 27):

- ◆ Finalize the class structure for the parsers.
- ◆ Refactor a pre-existing language parser to the new structure for testing.
- ◆ Make changes accordingly based on the test results.



→ Week 3 (June 27 - July 4):

- ◆ Add support for Rust.
- ◆ Add tests and documentation.

→ Week 4 (July 4 - July 11):

- ◆ Refactor the remaining language package parsers.
- ◆ Fix any bugs that may arise due to the refactor.

→ Week 5 (July 11 - July 18):

- ◆ Update and improve the tests for all the refactored parsers.
- ◆ Start working on a common API and modify the codebase appropriately.

→ Week 6 (July 18 - July 25):

- ◆ Finish working on the API.
- ◆ Cleanup the existing codebase to conform to the API.
- ◆ Create and update documentation regarding the API and other changes.

→ Week 7 (July 25 - August 1):

- ◆ Add support for R.
- ◆ Add tests and documentation.

→ Week 8 (August 1 - August 8):

- ◆ Add support for Ruby.
- ◆ Add tests and documentation.

→ Week 9 (August 8 - August 15):

- ◆ Add support for Golang.
- ◆ Add tests and documentation.

→ Week 10 (August 15 - August 22):

- ◆ Improve support for Python.
- ◆ Improve respective tests and documentation.

→ Week 11 (August 22 - August 29):

- ◆ Improve the overall testing suite for language parsers specifically.
- ◆ Add/iterate over documentation about the parsers along with how to generate their package lists.

→ Week 12 (August 29 - September 5):

- ◆ Add support for Windows package lists.
- ◆ Look into minor improvements for Linux package list and SBOM parsers.
- ◆ Add tests and documentation.

→ Final Week (September 5 - September 12):

- ◆ Buffer week.
- ◆ Fix bugs if there are any.
- ◆ Improve tests and documentation where they seem to be lacking.

→ Post GSoC:

- ◆ Continue contributing to the project by adding support for more languages and package managers.

Other Commitments:

- During mid July I will be having some minor university assessments as well as some pre scheduled hackathons because of which I would be traveling and wouldn't be able to put in enough time for the project, so in order to balance it out I would be dedicating more time in the other weeks so as to not hamper with the timelines.
- CVE Binary Tool is the only organization I am applying for.