CVE Binary Tool:
Building checkers/mappings for known package lists

About Me

Name: Muhammed Suhail A H
University: Government Engineering College, Palakkad
Program: Bachelor of Technology in Information Technology
Year: 3rd year
Email: msuhailbh07@gmail.com
GitHub: BreadGenie
Timezone: Indian Standard Time (GMT +5:30)

I'm a 3rd year student pursuing Information Technology at GEC Palakkad, Kerala. I've worked mostly with JavaScript/TypeScript for my projects. I set foot into Open Source by developing a Telegram bot. It was fun, gratifying and watching my friends and others using it and that made me realise how fulfilling it is to have something you build being used by others. I also love to empower people and that's the reason I took up the role of Program Manager for OpenHack'20 at FOSS-Cell-GECPKD, which was a program for programmers of all levels to set foot into Open Source. I'm also a member of RocketMeet, a meeting scheduling tool, where I worked on the client side.

I will be applying for GSoC at CVE Binary Tool because I'm interested in learning how a CLI tool works and how it's inner workings are connected to different components of the tool and I like how cve-bin-tool is used as a security measure in a wide variety of development areas. I would very much like to work on a real world python project and want to dive deep into the world of python development and I think this will be a great opportunity. And when I contributed to cve-bin-tool I got the opportunity to learn more about python and got excited at working on the project. I also want to refine my coding skills and best practices. Since I want to help people there's no better venue than a helpful tool used by many to avoid potential CVEs in their works. And it made me realise contributing to CVE Binary tool will help people out better and would make me much glad realising my work would be used by many.

I'm looking forward to working on cve-bin-tool in the future, as a contributor and maybe work on another project at cve-bin-tool for my final year capstone project.

Contributions

- **Pull Requests**
  - Add Checker for dnsmasq (#1076)
  - Add Checker for pspp (#1108)
  - CSV file of requirements.txt for CVE input (#1113)
  - Add Checker for ntp (#1127)
  - Add condensed downloads test (#1131)

- **Opened Issues**
  - New Checker: dnsmasq (#1075)
  - New Checker: pspp (#1107)
  - Unknown CVEs in output when there's at least one CVE in any of the packages and no CVEs in others (#1132)

Project Information

- **Sub-org Name**
  CVE Binary Tool

- **Project Abstract**
  Currently CVE Binary Tool’s binary scanning is slow and a software manifest of known packages can be used for quick scanning for CVEs as an early warning system. Here a database is built primarily of top packages from PyPI and the core packages of popular Linux distros like Ubuntu, Fedora, CentOS etc. and a parser that reads the packages from the list and maps them with the database. The checkers for these packages are created alongside.
  Also to improve user experience the creation of a backported fix checking utility can be made. It checks the packages with CVE in the scan and a database with the data including backported fix and provides an output if the CVE is fixed.

- **Detailed Description**

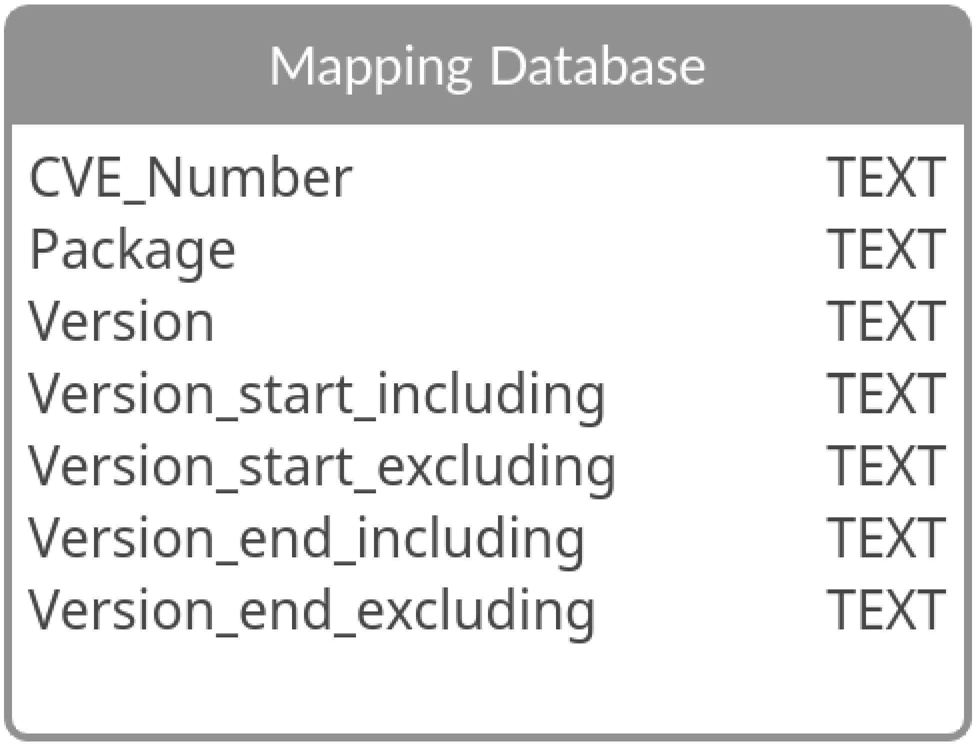
  Goal
  - Create databases for the popular packages from PyPI and core packages of popular Linux distros.
  - Create a parser to read the list of known packages and map them with the databases for a quick CVE scanning.
  - Create checkers for the above mapped packages.
  - Create a tool to prompt the user of the vendor-package pairs if the package isn't found in the mapping.
  - Create a tool to output the backported fixes in a package if there is any while scanning.

Implementation

Mapping Database

A usual binary CVE scan needs vendor-product pairs but here it may not be possible to get vendor info from requirements.txt or through listing installed packages of a distro. So a database for top packages of PyPI and core packages of Ubuntu, Fedora, CentOS etc. with their CVE Number are created for mapping. The other CVE details (Severity, CVE Score etc) can be queried from cve_severity table in cve.db.

The mapping database tables would follow the below schema



Mapping Database	
CVE_Number	TEXT
Package	TEXT
Version	TEXT
Version_start_including	TEXT
Version_start_excluding	TEXT
Version_end_including	TEXT
Version_end_excluding	TEXT

The schema is similar to the cve_range table in cve.db (except vendor). But here the package name could vary from what it is in the NVD database and is thus modified according to the list we will map with.

Parser

A parser is built for mapping the package list and the database.It either checks through requirements.txt or lists out installed packages in the system and reads through them for mapping.

We can handle this by providing a flag such as `-m or --manifest` for the cve-bin-tool and the parameters can be the path to the requirements.txt or left empty if it's to check the installed packages in the system.

For python packages we may not be able to acquire the version information from requirements.txt. So for that we could access the package details of installed packages in the system by executing `pip list` or `pip freeze` using the subprocess module and extract the product name and version of the installed packages by comparing it with the requirements.txt and use it to map on the created database.

Similarly for Ubuntu core packages we could run `dpkg-query -l` in the present working directory using the subprocess module and manipulate the output for mapping.

We could do the same with Fedora using `dnf list installed` and for CentOS using `yum list installed`.

Test Driven Development method is used for writing tests along with parser construction. The parsers will be documented along with the implementation.

Checkers

Checkers for all the possible packages compiled are made along with the construction of databases and parsers. Since there probably would be 100+ packages to write checkers for, they will be created according to the priority among the packages. There will be almost an equal number of checkers for PyPI top packages, Ubuntu, Fedora and CentOS core packages if there are enough packages with not so hard signatures.

Tool to recommend potential vendor-product pairs

When we do the quick scan there's a big possibility that we would miss scanning plenty of packages. So here we could output the potential vendor-product pair of the missed package by querying in cve.db and output them to the console. There could be chances for it to miss the mark. But it would definitely be useful for the users to know if a package escaped the mapping and it's vendor-product pair can be used by the user for further scanning.

Backported Fixes

There are certain packages with a version having CVEs but the fix has been backported due to issues like backward compatibility. So the existence of a functionality to notify the users if a certain package that was scanned doesn't have a CVE according to it's version since it's fixed, would be a helpful feature for the users so that they don't have to update those packages.

A possible way to implement this feature would be using RedHat CVE Database and similar CVE Databases that certain distros keep to track CVEs (scraping or downloading the Database if possible) and then use that data and compare with the packages that have CVEs according to the CVE Binary Tool.

- **Weekly Timeline**

  - **Pre GSoC**
    - Learn concepts related to the project in depth.
    - Read and discuss how to add the backport recognition functionality since the amount of work can’t be estimated right now.
    - Work on issues to get a grasp about the whole codebase or related parts.

  - **Community Bonding Period (May 17 - June 7)**
    - Bonding with the community actively.
    - Discussing and refining the project idea with mentors.

  - **1st Week and 2nd Week (June 7 - June 21)**
    - Creating the mapping Database for top PyPI packages.
    - Build a parser to read the listing of installed PyPI packages with the help of requirements.txt for mapping.
    - Create checkers for the above PyPI packages.
    - Create a tool to warn the users if a vendor-product pair is not found while mapping.
    - Document and write tests for the parser

  - **3rd Week and 4th Week (June 21 - July 5)**
    - Creating the mapping Database for core Ubuntu packages.
    - Build a parser to read the output of dpkg to map the packages.
    - Create checkers for the above packages.
    - Document and write tests for the parser

  - **5th Week and 6th Week (July 5 - July 19)**
    - Creating the mapping Database for core Fedora packages.
    - Build a parser to read the output of dnf to map the packages.

    - Create checkers for the above packages.
    - Document and write tests for the parser
    - Preparing code for evaluation.

- **7th Week and 8th Week (July 19 - August 2)**
    - Creating the mapping Database for core CentOS packages.
    - Build a parser to read the output of yum to map the packages.
    - Create checkers for the above packages.
    - Document and write tests for the parser

- **9th Week and 10th Week (August 2 - August 16)**
    - Implement a utility to check backported CVE fixes and report them in the output if the CVE is fixed in that specific version.

- **Buffer Week (August 16 - August 23)**
    - Fix bugs if there are any.
    - Improve tests.
    - Improve documentation.

- **Stretch Goals**
    - Adding core packages of other popular Linux Distros (SUSE, Arch Linux) and creating their checkers if the work is completed faster than I have projected.

      (This idea can be used as a stretch goal or as a post GSoC contribution)

Other Commitments

- I’ll have my final exam probably in July (not confirmed). So I won’t be able to put as much hours into the project at that time, so I’ll be working a bit longer in other weeks to balance it.
- CVE Binary Tool is the only organisation I’m applying for.