CVE Binary Tool: Improving Output for cve-bin-tool

Project information

Abstract

The project aims to improve the output of cve-bin-tool by adding different types of output options to the tool. The project will help users to generate machine-readable outputs as well as more extensive and detailed human-readable reports. The project will also improve “csv2cve” output by allowing users to generate outputs like that of “cve-bin-tool”. Formatting of JSON output will also be done according to the DFFML as they will be our first customer for JSON output.

If time permits project can include

- A coloured output for the console can be added to quickly differentiate between CVEs on the basis of products as well as severity.
- output CVE’s in some sort of table just like “Safety” (currently it’s a CSV dump)
- Internationalization of the tool to better support users.

Detailed description

The project deals with 3 main issues.

1. Machine Readable Output.
2. Generation of Descriptive Reports
3. Improved output of csv2cve.

Machine Readable Output:

CVE Binary Tool currently outputs 2 machine-readable outputs which are CSV and JSON. The project aims to reformat the CSV and JSON output to see if we can improve the CSV, JSON output. Doing sorting based on severity so that more severe CVE appears first or we can just omit that because sorting is a costly operation (needs discussion). As DFFML will be our first user for JSON output we can refactor the JSON output according to them.

Generation of Descriptive Reports:

Summary:

As discussed in #332 we want to generate more extensive reports from a command-line program which takes input in the form of machine-readable logs and we want the below-mentioned details to be included in that report.

- What files are affected
- What packages/versions were found
- What CVEs were found
- Some sort of colourized output to show you which CVEs are most scary by severity rating
- Maybe show more of the severity data? (We're using only the summary number but there's more there)
- Maybe a longer report with the short CVE description text
- Make the whole thing possible to generate from one of those machine-readable logs (see #262) so it could be done after the fact only as needed and you wouldn't have to re-run scans
- Maybe some graphs?
- Maybe some ability to track changes vs a previous log if one's available? (probably needs to go with machine-readable logs #262)

But as we can see from the above output requirements like printing files which are affected we need to save data during scanning. So we can add filename value to the machine-readable log or we can add a detailed descriptive report as an output type to the cve-bin-tool.

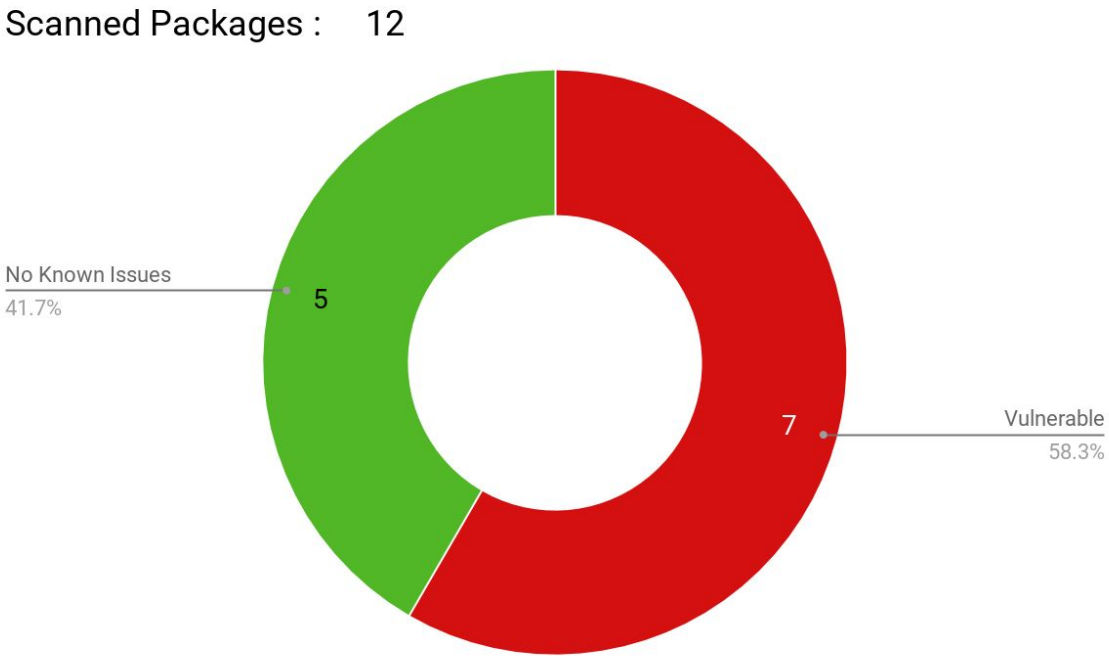
The motivation behind adding descriptive reports as an output type.

Suppose the user runs a scan and exports the output to the CSV or JSON format. We must update the database before scanning and when the user will provide the file to the descriptive report we must again update the database before we generate the output. This problem can be solved if the user is able to generate the report directly. So we can generate CSV or JSON as our intermediate file and then without leaving the program we can use that file to generate the Detailed reports. So in this way, we can generate Descriptive Reports from a Machine Readable Log(CSV, JSON) as well as by providing a “descriptive report” as an output format.

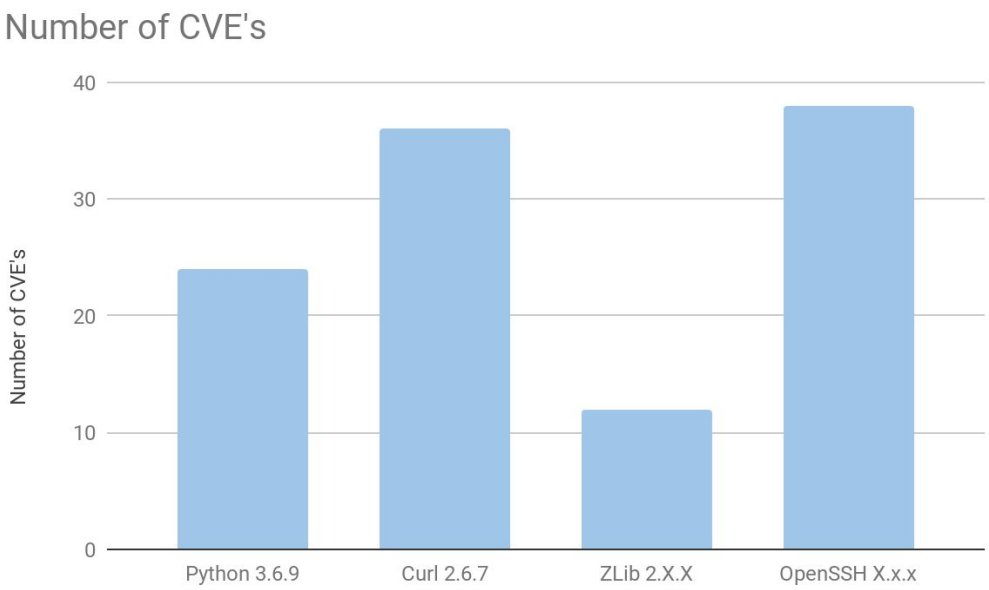
The project will generate descriptive reports of the scans in the form of HTML and help users to gather more insights regarding the project. Descriptive Reports will contain information in the form of graphs and summary reports for each CVE as well as the vendor/product pair.

The Basic information the report will generate is :

- Total Number of Products Found (including the ones that didn't have any CVE associated. (needs discussion: Should we also include the ones detected with unknown version?) This is a kind of similar to what @terriko was saying in #475)
    - On the basis of this, we can create a Pie chart showing infected products to that of total products. (-- maybe, also showing percentage?

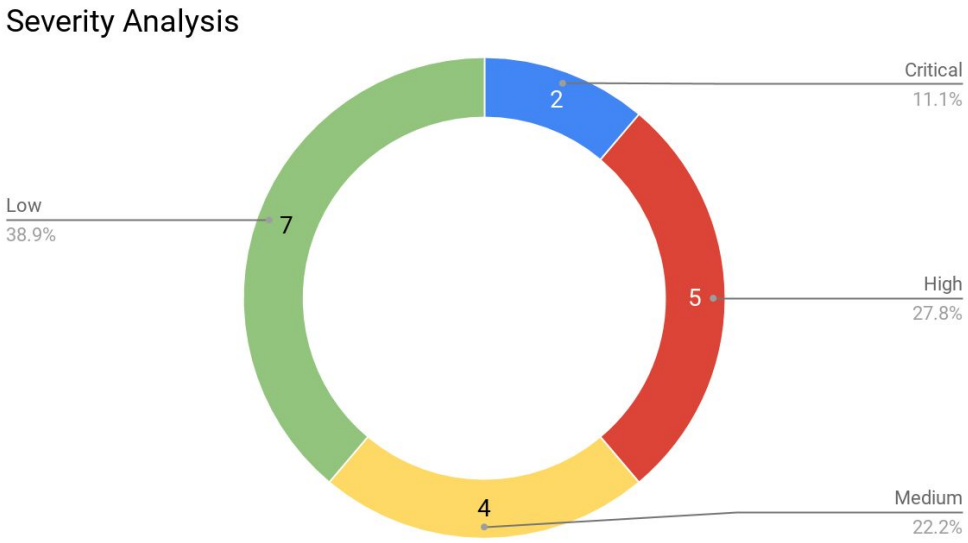


- The number of CVEs every product contains. This will be plotted as a column chart.
    - Example plot:



    -

- Then the report will start discussing each product one by one.

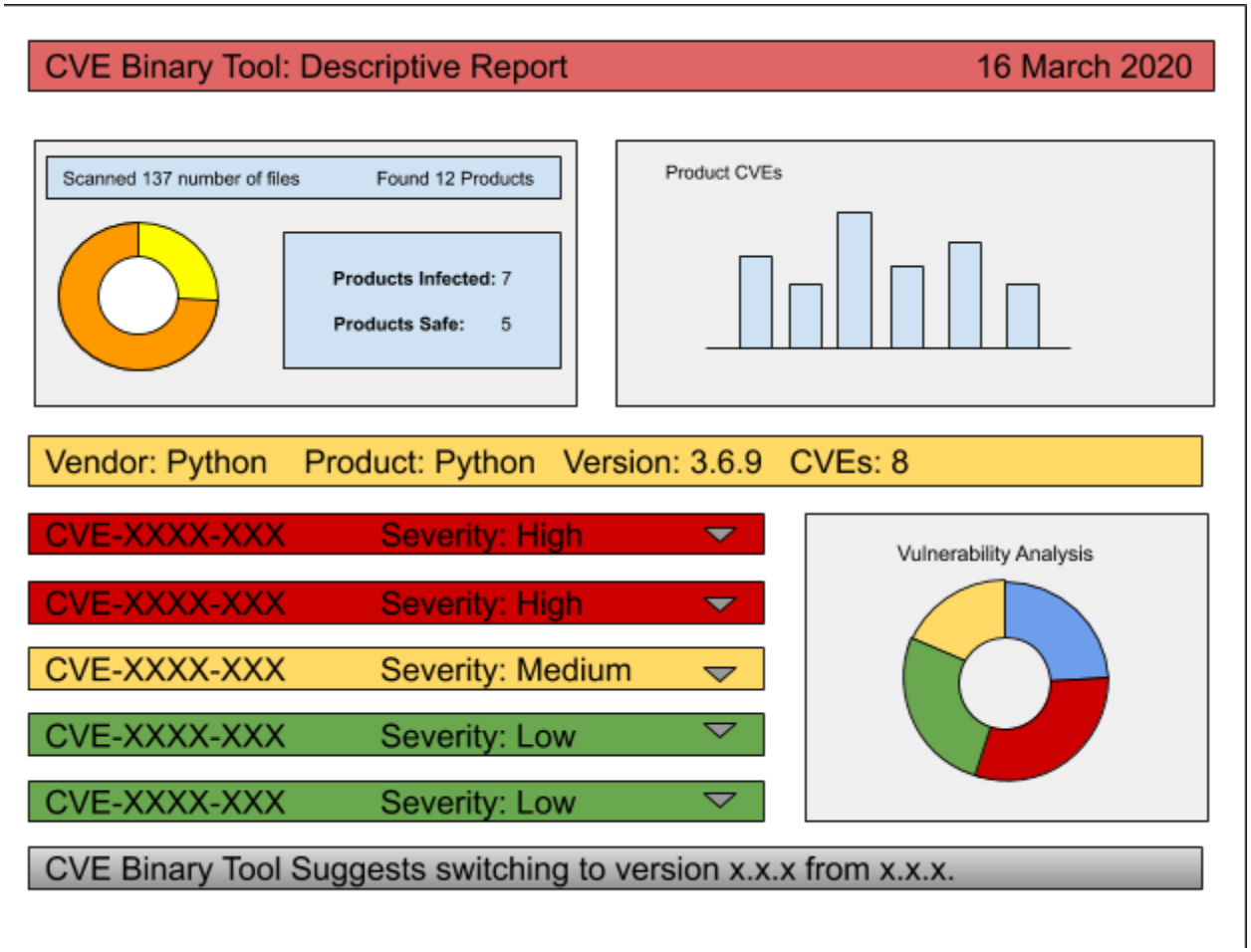


    - For each product, we will have a pie chart discussing the severity of CVE's
    - A description of every CVE in detail with its severity

- Suggestion to choose a version. Cve-bin-tool will be able to suggest the user a version. Maybe a version with less number of CVEs or a version with no vulnerability at all or a version having less number of high severity CVEs. (This also needs discussion)

Q) What should be the final output?

The following template gives a rough idea of how the descriptive report will look like. Description of every CVE can be seen by clicking on the arrow next to it.



Solution:

The idea mainly has three sub-issues

1. Generation of data
2. Graph generation
3. Template Generation

Generation Of Data: Here we want to scan the log and generate data related to the vendor/product. If a previous log exists we might want to differentiate data between the two logs.

Generation of Graphs: Here we can have multiple options but I'm not sure which one to choose.

1) Pygal: **pygal** is a dynamic SVG charting library written in python. Distributed Under LGPL-3.0
2) Chart.js: Simple, clean *and* engaging HTML5 based *JavaScript charts*. *Chart.js* is an easy way to include animated, interactive *graphs* on your website for free. Distributed Under MIT license

Generating template: The best practices often include using some templating engine like jinja 2 or something else. So this can also be figured out according to the needs.

Other issues include fetching new data every time we generate detailed descriptive reports. This will be overhead to the tool but I guess we can set the refresh time between 1-3 hours as I don't think the data changes much in such a small span of time. This will help users who want to generate multiple reports.

Other Features:

We can add a button in the HTML report to download the PDF version of the report.

Arguments for the program:

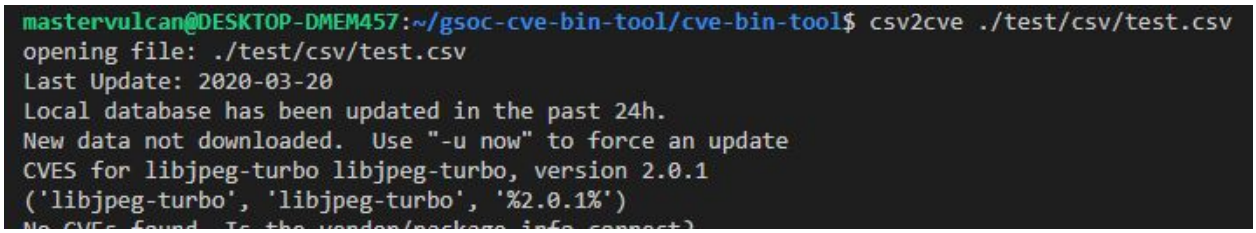
Descriptive report as a command-line program will receive arguments like

- File to be scanned - (Required)
- File format - (Required) option: CSV, JSON
- Filename of the output report.
- Database Updation - (Optional) options: now, daily, never etc..
- Previous Log - (Optional) Format needs to be the same as that of File to be scanned.

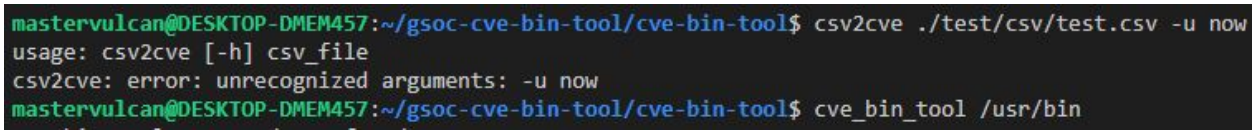
File to be scanned will be the file we want to generate the report from and the previous log if the user wants to compare the changes. The database argument will help the user to control how the database works.

Improved Output of csv2cve

Problem 1. Csv2cve misses some of the basic functionality like allowing users to update the database. Although it prints that “use -u now to update” but the tool does not handle the argument



(This figure shows csv2cve output “-u now”)



(This figure shows that csv2cve doesn't really accept that argument)

Currently, the csv2cve outputs CVE’s on its own and is lacking many features which cve-bin-tool offers such as the severity level and other information. Csv2cve also lacks the ability to produce machine-readable outputs like JSON and CSV. Adding this support will help users to generate a CSV(having additional information such as CVEs and severity rating) as well as JSON just by giving a CSV of vendor/product pair which can be also helpful for other services such as DFFML.

Solution

The project aims to solve all these issues stated in problem 1. Other than that the project aims for the unification of csv2cve output as well as cve-bin-tool output so that we can call OutputEngine.py right from csv2cve and generate whichever type of output we want like CSV,

JSON or any other which we might add in the future. This will be an efficient approach and a maintainable solution.

Other Ideas and Stretch Goals

If everything goes fine and I am left with some time I would love spending that implementing other ideas which are as follows.

- Adding support for triage data as discussed in #486
- Improving cve-bin-tool's console output. Currently, cve-bin-tool dumps the data in the form of CSV. But It would be better if we serve the output in a prettier format such as a table just like the "safety".
- A coloured output for the console can be added to quickly differentiate between CVEs on the basis of products as well as severity. This Idea on the top of "Safety" like output would be awesome.
- Internationalization of the tool can also be done. Which will help to increase our tool's user base and allow contributors from other regions to join our community and help us grow.

Code contribution

Improving Output Related Contributions:

- #410 - "Added OutputEngine.py for generating output"
- #375 - "unified string formatting using f-strings"
- #331 - "return unknown if the tool can't detect version"
- #448 - "JSON output_file formatted - more human-readable code"
- #318 - "Checkers Information printing Twice problem solved"

Other Contributions:

- #353 - "Added Python Checker"
- #339 - "Added Bluez Binary Test"

- #311 - “gnutls file test added for version 2.3.1”

Weekly timeline

- **Community Bonding** (May 4 - 31):
    - Setup Blog
    - Setting up of Communication
    - Learning More about community goals
    - Learning things which will be needed during the project

- **Week 1** (June 1- 7):
    - Learning about the requirements of DFFML so that we can modify our JSON output according to them.
    - Improving JSON output to better support DFFML
    - Improving CSV output. We can add sorting of output according to severity rating so that the CVEs with severity critical appears first followed by High, Medium and Low.
    - Improving Console output. Producing output in prettier format tables like that of safety.

      Expected Deliverables:

      Improved Machine-readable output as well as console output. Supporting DFFML with our JSON output.

- **Week 2** (June 8 - 14)
    - Improving the way csv2cve works

- Improving database updation output for csv2cve
- Learning the way csv2cve produce output
- Improving csv2cve so that it can support OutputEngine.py

  Expected Deliverables:

  Improved csv2cve working. The solution to Problem 1 as discussed in csv2cve detailed discussion. Csv2cve supporting OutputEngine

- **Week 3** (June 15 - 21)
  - Adding support to generate CSV, JSON, Console output from csv2cve.
  - Testing console output
  - Testing CSV output
  - Testing JSON output

    Expected Deliverables:

    Csv2cve now able to generate CSV, JSON, Console output.

- **Week 4** (June 22 - 28)
  - Start Planning Descriptive Report generation
  - Learning and planning how the data will flow
  - Figure out what kind of data will be needed
  - What technology stack or dependencies to use
  - Working on developing a command-line program as csv2cve.

    Expected Deliverables:

    Detailed Plan on how to tackle this giant problem. A command-line program which will generate the descriptive report.

- **Week 5** (June 29 - July 3) 1st Evaluation
    - Preparing the code for evaluation
    - Removing Bugs
    - Adding Fixes if needed
    - Testing the code

- **Week 6** (July 4 - 12)
    - Start working on the generation of Descriptive reports
    - Testing if the command line program can take input properly
    - Generating data needed for a detailed description. This includes modifying the Scanner data to include the filename which is being affected.
    - Adding functionality to control the Updating of database

      <u>Expected Deliverables:</u>

      Descriptive report program will be able to take input and generate the data according to that.

- **Week 7** (July 13 - 19)
    - Start working on the generation of graphs.
    - Generating Scanned Packages graph
    - Generating Product CVEs Graph
    - Generating Vulnerability Analysis graph
    - Testing of the graph

      <u>Expected Deliverables:</u>

      The tool will be able to generate graphs according to the data.

- **Week 8** (July 20 - 26)
    - Generating templates

- Generating CSS, JavaScript
- Testing the output

Expected Deliverables:

We can see the templates which will be used by the reports.

- **Week 9** (July 27 - 31) 2nd evaluation
  - Preparing the code for evaluation
  - Removing Bugs
  - Adding Fixes if needed
  - Testing the code

- **Week 10** (Aug 1 - 9)
  - Combining the Data + Graphs + Template
  - Generation of Full Descriptive Reports
  - Adding descriptive report as an output type.
  - Testing the output type

    Expected Deliverables:

    Full descriptive reports can be generated from the “descriptive reports” as well as from the “cve-bin-tool”.
- **Week 11** (August 10 - 16):
  - Buffer Period
  - Stretch Goals
  - Testing Code
  - Start preparing Documentation

- **Week 12** (August 17 - August 23) final submission
    - Fixing and removing Bugs
    - Documenting Code

- **The final week** (August 24 - August 31):
    - Preparation for Submission
    - Submitting Code

- **Post GSoC:** I will stick to the organisation and continue working on the project as there are many things which I might not be able to complete during GSoC and I want to work on them. The community is really good, especially mentors.

Other commitments

- I have participated in a national level contest SIH(***Smart India Hackathon***) and our team proposed a project for ***Indian Space Research Organisation*** **- ISRO** which has been selected by the judges. The SIH was to be held on 4th and 5th of April 2020 but due to ***COVID-19 (coronavirus)***, it has been postponed to 18-19 July. I will need a **4 day off** (17-20th July) for the SIH. But, I can hopefully give 2 hours to the project while travelling.

    - **Q) SIH is a two-day contest but why do I need 4 days off?**

      SIH for ISRO is at **Ahmedabad, Gujrat**. So, it will take approximately a day to travel there. Due to this, I will need 2 extra days.

About me

Full Name **Github UserName** **Gender**	Harmandeep Singh @SinghHrmn
Institute **Degree** **Major** **Expected Graduation Year**	Guru Nanak Dev University, Amritsar Bachelors of Technology Computer Science and Engineering 2021
Email **Mobile Number** **City, Country** **Postal Address**	
Time zone	Kolkata, INDIA (UTC +5:30)
Primary Language	English
Github **LinkedIn**	

I am a third-year student pursuing computer science. I'm skilled in python and C/C++. My major interest is python and I love to develop software with it. I have learned Django, Tkinter and made a few apps and web apps with them which can be viewed at my GitHub account. I love contributing to open source and have few projects of my own which are open source and I am interested to spend my summer doing what I love. I will spend 5 to 7 hours a day on the Project.

Previous Experience

I started to code in python from December 2017 and from that moment Python became my favourite language due to the two main reasons. 1) It was my first ever experience in programming. 2) Python was easy to understand. So after learning, I decided to participate in the (CSoC)CESS Summer of Code 2018, 2019 which is hosted every year by the technical team of CESS(Computer Engineering Students Society) of our university. CSoC is a miniature version of GSoC but with some rules changed. It is a 2-month long team contest and each team can choose from a given set of problems and file a proposal for the same. There are two evaluations in the CSoC period and each team needs to send a weekly report to the Judges. Our team created a students performance app as part of CSoC'18. The project was a GUI tool made with Tkinter and was to be used by CESS to store the list of upcoming events of CESS and even store the winners of previous events. We learned many new things while building that project. As my previous experience was really great I took part in CSoC'19 and this time our team developed a web app(Convert Master) based on Django which converts basic data formats like CSV, JSON, XML to each other and is hosted at https://convertmasterweb.pythonanywhere.com/. The app received 1st Prize at CSoC. Due to CSoC, I can claim that I have GSoC like experience but I can't claim that it's equivalent to GSoC as it is a solo project whereas CSoC was a team project. So I guess GSoC will be fun and challenging as it will open a wider learning scope and will further improve my skills so that I can better support and contribute to the cve-bin-tool community.