CVE Binary Tool

Improve CVE database structure and tracking of multiple reports

About Me:

Name:	Sahil
University:	University of Hyderabad
Github:	@imsahil007
Location:	India
Time Zone:	Indian Standard Time (GMT +5:30)
Primary Language:	English

Here's my portfolio

I am a 2nd year student pursuing Masters of Computer Applications (MCA) from University of Hyderabad, India. My classes are asynchronous leaving me enough time to work. I have already been contributing to the project since last year. So, I am quite prepared for this.

Abstract:

- To improve triage and tracking of reports:
  - A tool to append new scans to previous scan reports
  - A combining tool to merge different reports

- To improve structure of cvedb.py
  - Not reinitialize the database in case of timeout or some other problem
  - A way to fetch the modified NVD data rather than fetching complete CVE data using NVD's CVE Retrieval Support. This feature can be included for users who want to run scans very frequently .

Detailed Description:

- **To improve triage and tracking of reports:**

  Currently CVE Binary tool doesn't provide any medium to append intermediate reports.

  I propose to create a medium to save intermediate reports as json(s) using existing **output_json** and append them whenever required. A separate flag -a --append followed by a filename will keep on merging them together and display the updated output each time .

  Currently we store json output in the following dictionary format:

```
[
  {
```

```
        "vendor": $vendor,
        "product": $product",
        "version": "X.XX.XX",
        "cve_number": "CVE-20XX-1234",
        "severity": "MEDIUM",
        "paths": "/home/$prodcut-x.xx.xx_x86_64.rpm.tar.gz contains
/usr/sbin/$vendor",
        "remarks": "NewFound",
        "comments": ""
    },
]
```

Intermediate reports can be structured in similar way with some minor tweaks:

{

“timestamp(UTC)”: 2021-03-24T11:07:55Z,

“report”: [

Dictionary1,

..

Dictionary N

],

“type(optional)”: “backend”

}

- **A combine-report utility:**

  Users might also want to share reports from different teams (For example: Frontend and Backend) and merge them together. In this case, we can also attach some metadata like timestamp (if required) by users to get the source of the severity.

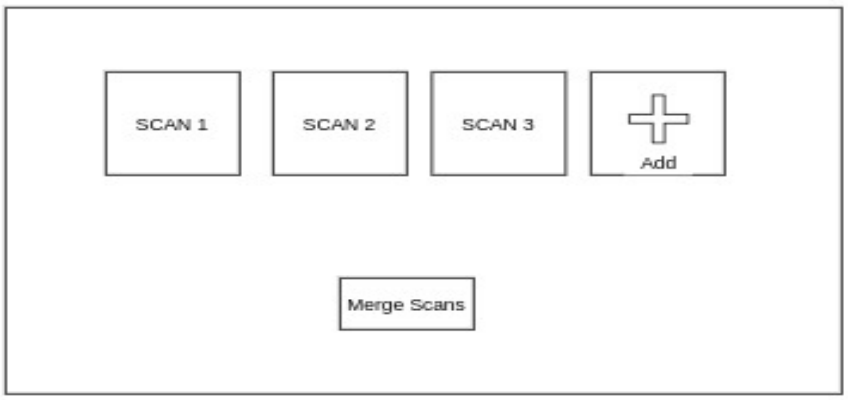
  We can combine these reports using the same cli tool mentioned above to save intermediate reports. Or we can create an independent -m --merge

Another approach can be to create an independent HTML page which will take an arbitrary number of json files and merge them after de-duplicating them using vanilla Javascript.

```
//De-duplicate scan dictionary
Array.prototype.unique = function() {
    var scan = this.concat();
    for(var i=0; i< scan.length; ++i) {
        for(var j=i+1; j< scan.length; ++j) {
            if(scan[i] === scan[j])
                scan.splice(j--, 1);
        }
    }

    return a;
};

//var scan1 = json loaded from scan1.json
//var scan2 = json loaded from scan2.json
var result = scan1.concat(scan2).unique();
/* Repeat same for n scans*/
```



Basic structure of HTML Page

- **Improve structure of cvedb.py**

  Our current database structure erases the database in case NVD scraping fails or a timeout occurs. We can avoid this by first creating a temporary cache directory and use this as our default database/cache only if no exception was caught while initializing it. Check

  This will also lay the foundation for supporting versioned cache directory.

- **Fetch only the modified CVE data**

With the addition of new NVD's CVE Retrieval API. We might need to fetch only the recent CVE data (which NVD updates after every 2 hours) rather than retrieving everything again. This will save us a lot of time and meaningless requests while loading the database to get the most recent CVEs.

With the addition of new NVDs CVE retrieval system - we will be able to fetch only the most recent and modified CVE record since our last request.

Two pairs of optional date parameters allow you to retrieve vulnerabilities based on when they were added to or modified in NVD, respectively.

Date parameters are in the form:

```
yyyy-MM-dd'T'HH:mm:ss:SSS z
```

The pubStartDate and pubEndDate parameters specify the set of CVE that were added to NVD (i.e., published) during the period. The modStartDate and modEndDate parameters specify CVE that were subsequently modified.

Check this link:

https://services.nvd.nist.gov/rest/json/cves/1.0?modStartDate=2021-04-01T00:00:00:000%20UTC-05:00

Code Contribution:

Merged Pull Requests

Issues Reported and Fixed

I have also gone through the whole source code a couple of times and I believe that I have well understood the working of the CVE Binary tool**.** So, I believe that I will be able to work comfortably during the GSoC period without much difficulties. Also, I plan to work on cve-bin-tool even after the GSoC period.

Weekly Timeline:

- **Community Bonding**

  Get a whole understanding of the requirements of CVE-Binary tool with the help of mentors. To create a list of separate issues that I will be working on.

- **Week 1 (June 7)**

  Add --append flag to save intermediate json reports using existing output_json. Also find if there can be better option to save these reports instead of json

- **Week 2 (June 14)**

  Create a combine report utility for saving and merging reports from different teams. Possibility of including other metadata along with timestamp and maybe some other user identification field.

- **Week 3 (June 21)**

  Create a HTML based combine report utility for saving and merging reports using vanilla javascript.

  Week 4 (June 28)

  Work on the UI of the above mentioned html page.

  Documentation for the above mentioned combine utility. Add some related tests for intermediate reports.

- **Week 5 (July 5)**

  Improve the structure of the cve database. Add some necessary datetime fields to the database. Make some changes to scrape data into a temporary database rather than default one.

- **Week 6 (July 12)**

  Delete/update the old database only when a new copy is created successfully.

- **Week 7 (July 19)**

  Create a utility flag to provide the user the option to fetch the modified NVD data rather than reinitializing the whole database every time using new CVE Data retrieval.

- **Week 8 (July 26)**

  Update documentation for database related changes. Write tests for checking when the database is updated using date range fetching of NVD data.

- **Week 9 (August 2)**

  Test the changes by combining reports from running different runs, updating the database on updated/modified NVD files multiple times to check for any edge cases.

- **Week 10 (August 9)**

  Take some feedback on the modified NVD update and check if more changes are required.

- **Final Week**
  Add and commit all changes in different pull requests and update the documentation and contributor's guide. Continue contributing even after the final week.