

SECURIN ASSESMENT

DATE: 2024-05-10

NAME: RITHESH S

NVD - CVE API

The CVE API is used to easily retrieve information on a single CVE or a collection of CVE from the NVD. Pls refer to the below NVD CVE documentation to get more information

TECH STACK

- python flask
- mongodb
- docker

FUNCTIONALITIES

- Retrieve CVE information from the CVE API for all the CVE's batch wise using **startIndex** and **perPage** get parameters. <https://services.nvd.nist.gov/rest/json/cves/2.0> store all the CVE INFORMATION in local **MONGO DB**.
- Retrieve CVE changes from the CVE HISTORY API and update the changes in the local db
- Removed **REJECTED** vulnerabilities from the database.
- Every 30 minutes it will query the HISTORY API for any changes and apply those changes to the local database
- All data are kept up-to-date with the server in an interval of 30 minutes. can be changed.
- Pagination on the **server side** is implemented
- Multiple Search filters are implemented can be using combined
 - cve id
 - year
 - baseScore

API ENDPOINT

ENDPOINT : **http://<domain>:<port>/**

/ and /cve/lists

- Lists out the CVE INFORMATION from the database in a clean table.

GET PARAMETERS

- ?page=

- datatype: `int`
- returns: returns the content of the page requested
- `?limit=`
 - datatype: `int`
 - returns: restricts the number of CVES in the page
 - options: 10, 50, 100
- `?year=`
 - datatype: `string`
 - returns: returns all the CVES in that given year
- `?lastmodified=`
 - datatype: `int`
 - returns: sorts the table with last modified in descending order and limits it to the number of days given
- `?lt=`
 - datatype: `float`
 - returns: BaseScore value lesser that the input
 - ref - `metrics.cvssMetricV2.cvssData.baseScore` or `metrics.cvssMetricV3.cvssData.baseScore`
- `?gt=`
 - datatype: `float`
 - returns: BaseScore value greater that the input
 - ref - `metrics.cvssMetricV2.cvssData.baseScore` or `metrics.cvssMetricV3.cvssData.baseScore`
- `?id=`
 - datatype: `string`
 - returns: Searches the CVE id either completely or partially. **REGEX MATCHING**

`/cve/lists/id/<id>`

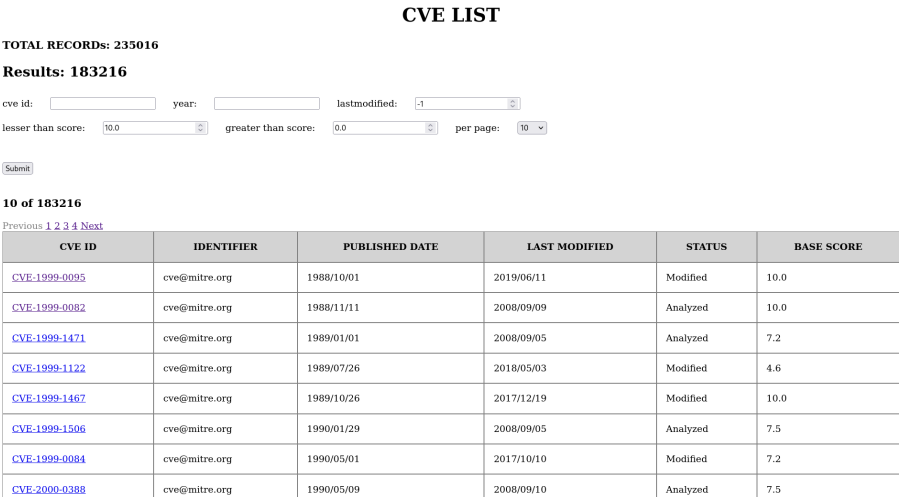
- returns the details of the specific CVE INFORMATION

UI

SEARCH

The previous requests are saved and previous values or loaded after page refresh

```
{
  "limit": limit,
  "year": year,
  "lastmodified": lastmodified,
  "lt": lt,
  "gt": gt,
  "id": id
}
```



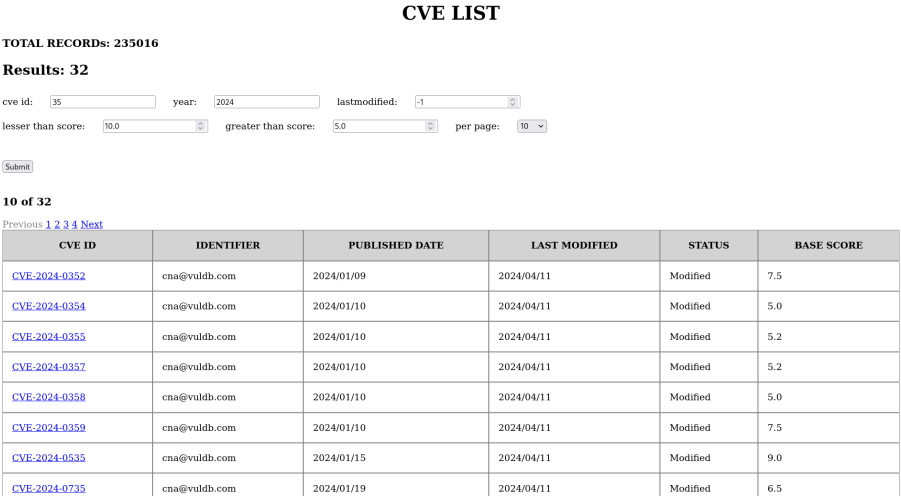


Figure 3: CVE ID CONTAIN 35 AND IN YEAR 2024 AND BASESCORE GREATER THAN 5.0

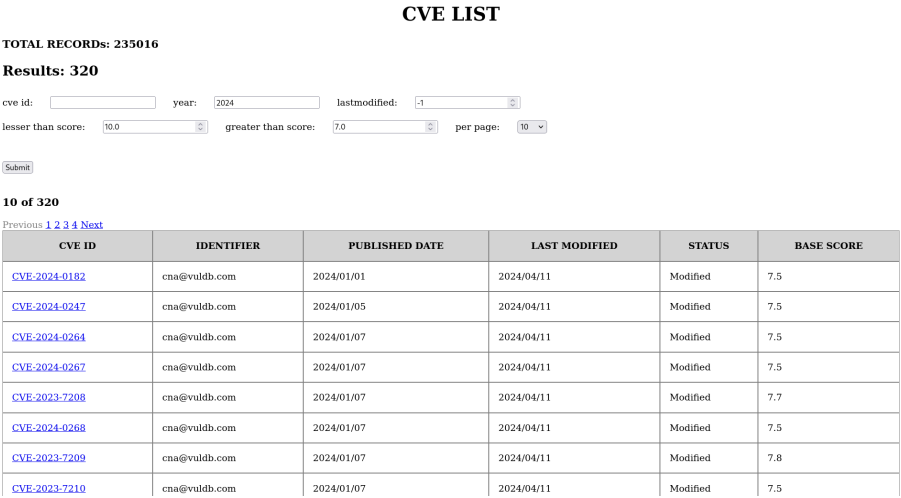


Figure 4: PAGINATION EXAMPLE

```

util.py > @ main
26     start_index += per_page
27 else:
28     time.sleep(5)
29     continue
24     (if data['resultsPerPage'] == 0:
25         break
26     write_dbdata['vulnerabilities'])
27
28 def main():
29     start_index = 0
30     per_page = 2000
31     while True:
32         #debug
33         status_code, data = get_data(start_index, per_page)
34         #print(status_code, ":", start_index, ":", (start_index / 240025) * 100)
35         (if status_code == 200:
36             start_index += per_page
37         else:
38             time.sleep(5)
39             continue
40         (if data['resultsPerPage'] == 0:
41             break
42         write_dbdata['vulnerabilities'] = [{"id": "cve"}]
43         write_dbdata['vulnerabilities'])
44
45 if __name__ == "__main__":
46     main()
47     while True:
48         clean_db()
49         time.sleep(30 * 60) # sleeps for 30 minutes and updated the cves list
50         handle_update()
51
52 v main 2 2

```

Figure 5: util.py sample code

CODE IMPLEMENTATION

On first time run the util.py it will populate the database by querying the CVE api endpoint After the database is synced. It will delete all the REJECTED vulnerabilities. It will then wait for 30 minutes or specified time. to then query the history api and make changes again The last queried time is written to config.json and history after that time is then queried

```

payload = {
    "published": {
        "$regex": year
    },
    "$or": [
        {"metrics.cvssMetricV2.cvssData.baseScore": {"$gte": gt, "$lte": lt}},
        {"metrics.cvssMetricV3.cvssData.baseScore": {"$gte": gt, "$lte": lt}}
    ],
    "id": {
        "$regex": id
    }
}

```

This sets the filter query paramter in the mongodb based on the input args from the GET parameters

```

app.py
1 inputs = {
2     "limit": limit,
3     "year": year,
4     "lastmodified": lastmodified,
5     "lt": lt,
6     "gt": gt,
7     "id": id
8 }
9
10 #datas = collection.find()#.limit(10)
11 payload = {
12     "published": {
13         "$regex": year
14     },
15     "$or": [
16         {"metrics.cvssMetricV2.cvssData.baseScore": {"$gte": gt, "$lte": lt}},
17         {"metrics.cvssMetricV3.cvssData.baseScore": {"$gte": gt, "$lte": lt}}
18     ],
19     "id": {
20         "$regex": id
21     }
22 }
23 #datas = collection.find(payload).limit(10)
24
25 if lastmodified != -1:
26     datas = collection.find(payload).sort("lastModified", -1).skip((page - 1) * limit).limit(lastmodified)
27 else:
28     datas = list(datas)
29     datas = collection.find(payload).skip((page - 1) * limit).limit(limit)
30
31 # Determine if there are more pages
32 search_total = collection.count_documents(payload)
33 has_next = (page * limit) < search_total
34 has_prev = page > 1
35
36 return render_template("table.html", datas=datas, total=search_total, search_total=search_total, inputs=inputs, page=page, has_next=has_next, has_prev=has_prev )
37
38 @app.route('/cve/lists/id/<cve_id>')
39 def cve_id(cve_id):
40     data = collection.find_one(cve_id)
41     if data == None:
42         return "NO CVE IN THAT ID"
43     return render_template("single.html", data=data)
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Figure 6: app.py sample code

WORKFLOW

LIST VIEW

You can see the list of CVES in / or /cve/lists endpoint.

You can **chain multiple search paramters** together to have a strong searching capabilities

Results per page can be configured

Results are paginated and previous and next page pagination are done server side

SINGLE VIEW

View specific details about the CVE

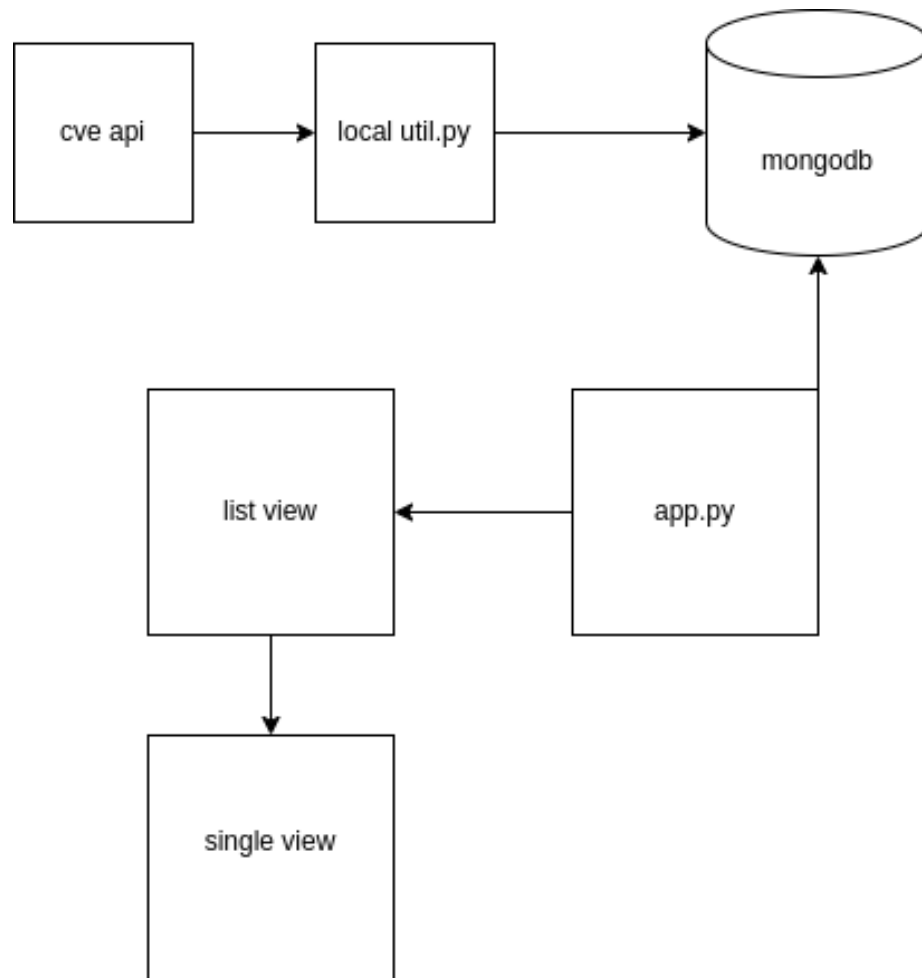


Figure 7: workflow architecture diagram

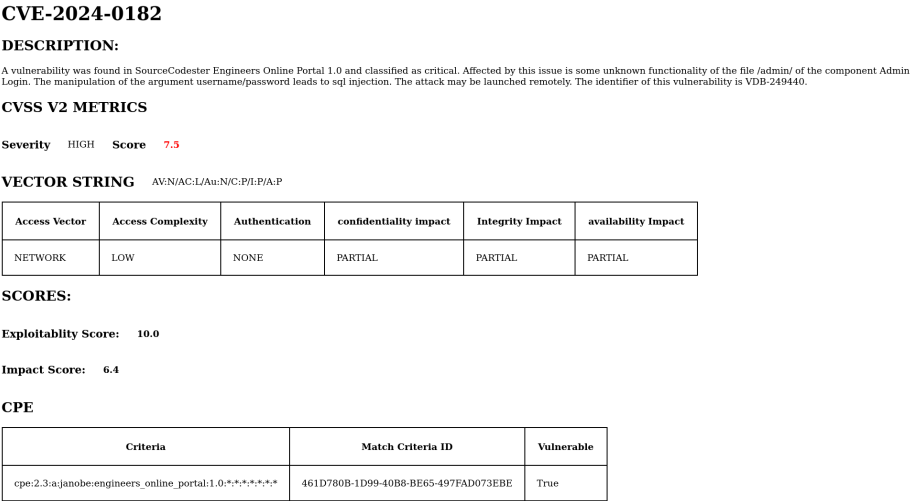


Figure 8: SPECIFIC CVE