Securin Assessment

Date:16.05.2024
Name: Vasanth M

Overview:

The CVE API is used to easily retrieve information on a single CVE or a collection of CVE from the NVD and Visualizing the api data from a local database.

Tech Stack:

Express: For handling server requests and responses.
Axios: For fetching data from the api
Mongoose: For connecting to and interacting with MongoDB.
Ejs: For rendering views.

Functionalities Implemented:

1. Fetched CVE data in batches from the NVD's CVE API:https://services.nvd.nist.gov/rest/json/cves/2.0 The data fetching is controlled by offset parameters using startIndex and resultsPerPage.The fetched data is stored in a collection of the local MongoDB database.
2. Retrieve CVE changes from the CVE HISTORY API and if there are any changes update in the local db.
3. Removed REJECTED vulnerabilities from the database for having clean data.
4.After every 30 minutes,call the history api and for any changes in the CVE data and update it in the Mongodb collection to maintain data integrity.
5.Server side pagination is implemented.
6.Options for results per page can be chosen.The options are 10,50,100.
7.CVE data are sorted in ascending order of their published date.
8.All the CVE rows have a link which links to their respective details page.
9.The information about each CVE is displayed in their respective page.
10.Filters can be applied on these following parameters:

- CVE id
- Score
- Year
- N last modified data

Endpoints

1. **/cves/list**

   This endpoint fetches CVE data from the database.

   Request Type: GET

   Query Parameters:

   page: (integer) Specifies the page number of the results to return. Default is 1.
   perPage: (integer) Limits the number of CVEs returned in a single page. Default is 10.
   year: (string) Filters the CVEs to return only those from the specified year.
   lastModified: (integer) Sorts the table with last modified in descending order and limits it to the number of days given.
   lt: (float) Returns CVEs with a BaseScore value less than the input. Refers to metrics.cvssMetricV2.cvssData.baseScore or metrics.cvssMetricV3.cvssData.baseScore.
   gt: (float) Returns CVEs with a BaseScore value greater than the input. Refers to metrics.cvssMetricV2.cvssData.baseScore or metrics.cvssMetricV3.cvssData.baseScore.

2. **/idResult**

   This endpoint fetches a specific CVE document from the database based on the CVE id.

   Request Type: GET

   Query Parameters:

   SearchId: (string) The CVE id to search for.

3. **/cves/list/:cveid**

   This endpoint fetches a specific CVE document from the database based on the CVE id provided in the URL and returns a detailed view of that particular CVE id.

   Request Type: GET

   URL Parameters:

   cveid: (string) The CVE id to search for.

UI

Total Records:183216

ID Search

Filter

Records Per Page 10

score gt score lt Last Modified Year

Apply

CVE DATA

Page 1 of 18322

CVE ID	Identifier	Published Date	Last Modified Date	Status	Score
CVE-1999-0095	cve@mitre.org	01 October 1988	11 June 2019	Modified	10
CVE-1999-0082	cve@mitre.org	11 November 1988	09 September 2008	Analyzed	10
CVE-1999-1471	cve@mitre.org	01 January 1989	05 September 2008	Analyzed	7.2
CVE-1999-1122	cve@mitre.org	26 July 1989	03 May 2018	Modified	4.6
CVE-1999-1467	cve@mitre.org	26 October 1989	19 December 2017	Modified	10
CVE-1999-1506	cve@mitre.org	29 January 1990	05 September 2008	Analyzed	7.5
CVE-1999-0084	cve@mitre.org	01 May 1990	10 October 2017	Modified	7.2
CVE-2000-0388	cve@mitre.org	09 May 1990	10 September 2008	Analyzed	7.5
CVE-1999-0209	cve@mitre.org	14 August 1990	09 September 2008	Analyzed	5
CVE-1999-1391	cve@mitre.org	03 October 1990	05 September 2008	Analyzed	7.2

1 2 3 4 5 Next

Fig 1.Main page

Total Records:183216

ID Search

Filter

Records Per Page 10

score gt score lt Last Modified Year

Apply

CVE DATA

Page 5 of 18322

CVE ID	Identifier	Published Date	Last Modified Date	Status	Score
CVE-1999-1395	cve@mitre.org	17 November 1992	31 October 2009	Modified	7.2
CVE-1999-1306	cve@mitre.org	10 December 1992	05 September 2008	Analyzed	7.5
CVE-1999-1466	cve@mitre.org	10 December 1992	05 September 2008	Analyzed	7.5
CVE-1999-1021	cve@mitre.org	30 December 1992	10 October 2017	Modified	7.2
CVE-1999-0312	cve@mitre.org	13 January 1993	17 August 2022	Modified	5
CVE-1999-1507	cve@mitre.org	03 February 1993	10 October 2017	Modified	7.2
CVE-1999-1218	cve@mitre.org	18 February 1993	19 December 2017	Modified	2.1
CVE-1999-1312	cve@mitre.org	24 February 1993	19 December 2017	Modified	7.2
CVE-1999-1216	cve@mitre.org	22 April 1993	19 December 2017	Modified	7.5
CVE-1999-1162	cve@mitre.org	24 May 1993	05 September 2008	Analyzed	6.4

First Previous 1 2 3 4 5 Next

Fig 2.Pagination

Total Records:1038

ID [ ] Search

Filter

Records Per Page [10]
score gt [6.5] score lt [10.0] Last Modified [10] Year [2004]
Apply

CVE DATA

Page 1 of 104

CVE ID	Identifier	Published Date	Last Modified Date	Status	Score
CVE-2004-0285	cve@mitre.org	23 November 2004	23 April 2024	Analyzed	7.5
CVE-2004-2343	cve@mitre.org	31 December 2004	11 April 2024	Modified	7.2
CVE-2004-2339	cve@mitre.org	31 December 2004	11 April 2024	Modified	7.2
CVE-2004-0005	cve@mitre.org	03 March 2004	16 February 2024	Analyzed	7.5
CVE-2004-0119	cve@mitre.org	01 June 2004	15 February 2024	Analyzed	7.5
CVE-2004-0389	cve@mitre.org	01 June 2004	15 February 2024	Analyzed	7.8
CVE-2004-2154	secalert@redhat.com	31 December 2004	15 February 2024	Analyzed	7.5
CVE-2004-0213	cve@mitre.org	06 August 2004	14 February 2024	Analyzed	7.2
CVE-2004-0510	cve@mitre.org	23 December 2004	14 February 2024	Modified	7.2
CVE-2004-0507	cve@mitre.org	18 August 2004	14 February 2024	Modified	10

1 2 3 4 5 Next

Fig 3.Filtered search

Total Records:38

ID [0095] Search

Filter

Records Per Page [10]
score gt [ ] score lt [ ] Last Modified [ ] Year [ ]
Apply

CVE DATA

Page 1 of 4

CVE ID	Identifier	Published Date	Last Modified Date	Status	Score
CVE-1999-0095	cve@mitre.org	01 October 1988	11 June 2019	Modified	10
CVE-2000-0095	cve@mitre.org	24 January 2000	10 September 2008	Analyzed	5
CVE-2001-0095	cve@mitre.org	12 February 2001	30 October 2018	Modified	1.2
CVE-2002-0095	cve@mitre.org	25 March 2002	05 September 2008	Analyzed	7.5
CVE-2003-0095	cve@mitre.org	03 March 2003	18 October 2016	Modified	10
CVE-2004-0095	cve@mitre.org	17 February 2004	10 October 2017	Modified	5
CVE-2005-0095	cve@mitre.org	15 January 2005	11 October 2017	Modified	5
CVE-2006-0095	cve@mitre.org	06 January 2006	19 October 2018	Modified	2.1
CVE-2007-0095	cve@mitre.org	05 January 2007	29 July 2017	Modified	5
CVE-2008-0095	cve@mitre.org	08 January 2008	15 October 2018	Modified	5

1 2 3 4 5 Next

Fig 4.Search by a part of ID

CVE-1999-1122

Description:

Vulnerability in restore in SunOS 4.0.3 and earlier allows local users to gain privileges.

CVSS V2 Metrics

Severity: MEDIUM

score: 4.6

vectorString AV:L/AC:L/Au:N/C:P/I:P/A:P

Access Vector	Access Complexity	Authentication	Confidentiality Impact	Integrity Impact	Availability Impact
LOCAL	LOW	NONE	PARTIAL	PARTIAL	PARTIAL

Scores:

exploitabilityScore: 3.9

impactScore: 6.4

CPE

Criteria	Match Criteria Id	Vulnerable
cpe:2.3:o:sun:sunos:*:*:*:*:*:*:*:*	8ABF98B1-CBDE-4E12-8F3F-9D9E22E72236	true
cpe:2.3:o:sun:sunos:4.0:*:*:*:*:*:*:*	2839042D-7706-4059-B069-72E36297ECEB	true
cpe:2.3:o:sun:sunos:4.0.1:*:*:*:*:*:*:*	3791C6C1-2B30-4746-B4D5-A728914C3589	true

Fig 5.Detailed view of a CVE ID

Workflow:

Data Population: Start by populating your MongoDB database with CVE (Common Vulnerabilities and Exposures) data. You fetch the data from the NVD (National Vulnerability Database) using their CVE API. The data is fetched in batches, controlled by an offset parameter. For each vulnerability in the response,update the corresponding document in the MongoDB collection. If the document doesn't exist, it inserts a new one.This process continues until there's no more data to fetch.

Data Cleansing: After populating the database,perform a data cleansing operation where you remove all the rejected CVEs from the database.

Data Serving: Start the Express.js server that serves the CVE data. I have several routes set up for this:
1.The /cves/list route displays a list of CVEs.The data can be filtered based on several query parameters such as year, last modified date, and score range.
2.The /idResult route displays a list of CVEs that match a specific ID. The data is fetched from the MongoDB database and sent to the mainTable view to be rendered.
3.The /cves/list/:cveid route displays detailed information about a specific CVE. The data is fetched from the MongoDB database and sent to the subTable view to be rendered.

Data Updating: Every 30 minutes,fetch updates from the NVD's CVE history API and update the MongoDB database accordingly. For each change in the response, if the event name is "CVE Rejected", delete the corresponding CVE from the MongoDB collection. Otherwise, it fetches the CVE data from the CVE API and updates the corresponding document in the MongoDB collection.

Code Samples:

1.Populate DB and Cleaning:

```
async function populate(offset) {
    const apiUrl = 'https://services.nvd.nist.gov/rest/json/cves/2.0';
    const resultsPerPage = 2000;
   let offset = offset;
    try {while (true) {
            const response = await axios.get(apiUrl, {params: {startIndex: offset, resultsPerPage}});
            const currentPageData = response.data;
            const vulnerabilities = currentPageData.vulnerabilities || [];
            for (const { cve } of vulnerabilities) {
            const cveId = cve.id;
            await mongoose.connection.db.collection('Secure').updateOne({ _id: cveId }, { $set: cve }, { upsert: true })
            }
           offset += resultsPerPage;
           await new Promise(resolve => setTimeout(resolve, 10000));
        }
    } catch (error) {console.error('Error fetching data:', error.message);}
}
populate(0);
async function cleansing(){
    await mongoose.connection.db.collection('Secure').deleteMany({ vulnstatus: "Rejected" });
}
cleansing();
```

2.Main route(/cves/list):

```
app.get('/cves/list', async (req, res) => {
    const page = req.query.page;
    const perPage = parseInt(req.query.perPage) || 10;
    const year = req.query.year || "";
    const lastmodified = parseInt(req.query.lastModified) || -1;
    const lt = parseFloat(req.query.lt) || 10.0;
    const gt = parseFloat(req.query.gt) || 0.0;
    const id = req.query.SearchId || "";
    try {
        const totalCount = await mongoose.connection.db.collection('Secure').countDocuments();
        const totalPages = Math.ceil(totalCount / perPage);
        const offset = (page - 1) * perPage;
    const query = {
          "_id":{$regex:id},
          "published": {"$regex": year},
          "$or": [
            { "metrics.cvssMetricV2.cvssData.baseScore": { "$gte": gt, "$lte": lt } },
            { "metrics.cvssMetricV3.cvssData.baseScore": { "$gte": gt, "$lte": lt } }
        ]
       };
       let cve;
       if(lastmodified==-1){ cve = await mongoose.connection.db.collection('Secure').find(query).sort({published:1}).skip(offset).limit(perPage).toArra
       }
       else{
         cve = await mongoose.connection.db.collection('Secure').find(query).sort({lastModified:-1}).skip(offset).limit(lastmodified).toArray();
       }
      let tot = await mongoose.connection.db.collection('Secure').countDocuments(query);
      res.render('mainTable', {lastModified:lastmodified,lt,gt,year,perPage,total:tot,cve, totalPages, currentPage: page,formatDate:formatDate});
    } catch (error) {
        console.error("Error:", error);
}});
```

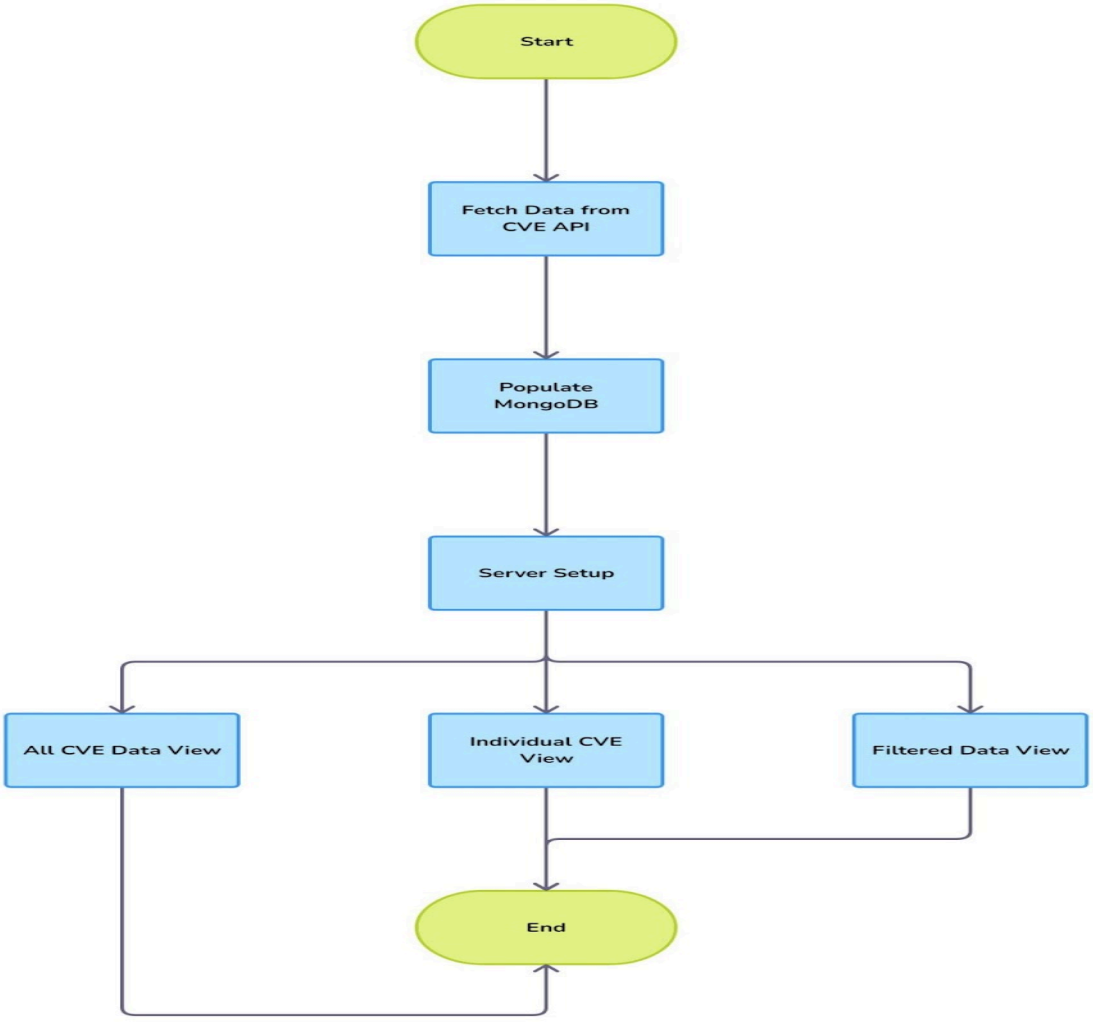
3.Searching route(/idResults):

```
app.get("/idResult", async (req, res) => {
    const id = req.query.SearchId;
    const perpage= 10;
    const query={
        "_id":{$regex:id}
    };
   const cve = await mongoose.connection.db.collection('Secure')
        .find(query)
        .limit(10)
        .toArray();
    let tot = await mongoose.connection.db.collection('Secure').countDocuments(query);
        const totalPages = Math.ceil(tot / perpage);
    res.render('mainTable', {cve, total: tot, totalPages: totalPages,perPage:10,year:"",
    currentPage: 1, formatDate: formatDate,gt:0,lt:10,lastModified:-1});
});
```

4.Periodic update:

```
async function updateDB(offset){
 const apiUrl = 'https://services.nvd.nist.gov/rest/json/cvehistory/2.0';
 let off = offset;
  try {while (true) {
            const response = await axios.get(apiUrl, { params: {startIndex: off,resultsPerPage:1}});
            const historyPageData = response.data;
            const histLen = historyPageData.totalResults;
            if(histLen!==localLen){
                const newcount = histLen - localLen;
                const newResponse = await axios.get(apiUrl, { params: {startIndex: newcount,resultsPerPage:localLen}});
                const newhistoryres = newResponse.data;
                for(let j=0;j<newcount;j++){
                    const change = newhistoryres.cveChanges[j].change;
                    if (newhistoryres.cveChanges[j].eventName === "CVE Rejected") {
                        await myColl.findOneAndDelete({ id: change.cveId }) }
                    else{
const cveDetailsResponse = await axios.get(`https://services.nvd.nist.gov/rest/json/cves/2.0?cveId=${change.cveId}`);
                        const cveDetailsResult = await cveDetailsResponse.json();
                        console.log(cveDetailsResult.vulnerabilities[0]);
                        const item = cveDetailsResult.vulnerabilities[0].cve;
                        item._id = cveDetailsResult.vulnerabilities[0].id;
                        await myColl.updateOne({ _id: cveData.id }, { $set: cveData }, { upsert: true });
                        await new Promise(resolve => setTimeout(resolve, 100000));}}
                localLen+=newcount;
                 console.log(`Updated DB with ${newcount} records`);}
        else{console.log("DB is in sync");}}}
catch (error) {console.error("An error occurred:", error);}}
```

Flow chart:


Start
Fetch Data from CVE API
Populate MongoDB
Server Setup
All CVE Data View
Individual CVE View
Filtered Data View
End