

CVE 漏洞分析系统

1. 分析 Github-cvelistV5 仓库中的 CVE 漏洞数据, 读取 json 文件中的 CVSS Score, CWE ID, references 信息;
2. 开发一种 facade 设计模式的 git repo 分析模块, 提取 reference 中的提到的 github 仓库的编程语言信息
3. 将这些信息存入 SQL 数据库

1. 调用 GitHub API 获得仓库语言信息

```
In [ ]: import requests
import json

def get_repo_lang(url):
    # 发送GET请求获取JSON文件内容
    if 'gist.github' in url:
        url = 'https://api.github.com/gists/'+url.split('/')[ -1]
    else:
        index = url.split('/').index('github.com')
        url = 'https://api.github.com/repos/' + \
            url.split('/')[index+1]+'/' +url.split('/')[index+2]
    response = requests.get(url)
    # 检查请求是否成功
    if response.status_code == 200:
        json_content = response.json()
        # 递归遍历每一层,找到所有字段名为 'Language' 的值

        def get_value(obj, key):
            arr = []

            def get(obj, key):
                if isinstance(obj, dict):
                    for k, v in obj.items():
                        if isinstance(v, (dict, list)):
                            get(v, key)
                        elif k == key:
                            arr.append(v)
                elif isinstance(obj, list):
                    for item in obj:
                        if isinstance(item, (dict, list)):
                            get(item, key)
            get(obj, key)
            return arr
        # 获取所有字段名为 'Language' 的值
        lang_list = get_value(json_content, 'language')
        # print('Language List:', lang_list)
        return lang_list[0] if lang_list else None
    else:

        print('Request failed with status code:', response.status_code)
```

```
# https://github.com/pterodactyl/wings/security/advisories/GHSA-p8r3-83r8-jwj5
# print(get_repo_lang('https://gist.github.com/xbz0n/674af0e802efaafe90d2f67464
print(get_repo_lang(
    'https://github.com/pterodactyl/wings/security/advisories/GHSA-p8r3-83r8-jwj
```

Go

```
In [ ]: url1 = 'https://github.com/pterodactyl/wings/security/advisories/GHSA-p8r3-83r8-
index = url1.split('/').index('github.com')
url1 = 'https://api.github.com/repos/' + \
    url1.split('/')[index+1]+'/' + url1.split('/')[index+2]
url1
```

```
Out[ ]: 'https://api.github.com/repos/pterodactyl/wings'
```

从单个 JSON 中提取 CVE 漏洞相关信息

1. CVE 编号
2. CWE 编号
3. 漏洞名称
4. 所涉及到的语言
5. 漏洞危害评分
6. 引用链接

```
In [ ]: import os
import time
import json
import random

THRESHOLD = 10 # 设置一个测试阈值

def get_cve_from_json(path):
    with open(path, 'r', encoding='utf8') as f:
        info = json.load(f)
        cve_id = info.get('cveMetadata', {}).get('cveId', None)
        cwe_id = info.get('containers', {}).get('cna', {}).get(
            'problemTypes', [{}])[0].get('descriptions', [{}])[0].get('cweId')
        title = info.get('containers', {}).get('cna', {}).get('title', None)
        language = None
        score = info.get('containers', {}).get('cna', {}).get('metrics', [{}])[
            0].get('cvssV3_1', {}).get('baseScore', None)
        references = info.get('containers', {}).get(
            'cna', {}).get('references', [])
        urls = []
        for i in references:
            url = i.get('url', None)
            # 如果Url中存在github
            try:
                if 'github' in url:
                    # 随机睡眠0.5~1.2s
                    time.sleep(random.uniform(0.5, 1.2))
                    language = get_repo_lang(url)
                    pass
            except:
                pass
            urls.append(url)
```

```

        # 将urls列表中的每一个元素用;号连接成一个字符串
        urls = '; '.join(urls)
        return cve_id, cwe_id, title, language, score, urls

print(get_cve_from_json(r'cvelistV5\cves\2023\0xxx\CVE-2023-0830.json'))

```

('CVE-2023-0830', 'CWE-78', 'EasyNAS backup.pl system os command injection', None, 6.3, '')

SQLite3 数据库建表

```

In [ ]: import sqlite3

# 连接到SQL数据库
conn = sqlite3.connect('cve_database.db')
cursor = conn.cursor()

# 创建表格
cursor.execute('''CREATE TABLE IF NOT EXISTS CVE_INFO
                  (CVE_ID TEXT PRIMARY KEY NOT NULL,
                   CWE_ID TEXT,
                   TITLE TEXT,
                   LANGUAGE TEXT,
                   Score DOUBLE,
                   REF TEXT);''')
conn.commit()
conn.close()

```

```

In [ ]: conn = sqlite3.connect('cve_database.db')
        cursor = conn.cursor()
        # 删除数据库Table中所有条目
        # cursor.execute('DELETE FROM CVE_INFO')
        conn.commit()
        conn.close()

```

添加 2023 年的所有 CVE 漏洞信息至数据库中

1. 数据库增查函数实现
2. 递归遍历 2023 年文件夹下的所有 JSON 文件
3. 调用先前的信息提取函数

```

In [ ]: # 插入CVE信息
        conn = sqlite3.connect('cve_database.db')
        cursor = conn.cursor()

        def insert_cve_info(cve_id, cwe_id, title, language, cvss_score, references):
            insert_query = '''
            INSERT INTO CVE_INFO (CVE_ID, CWE_ID, TITLE, LANGUAGE, SCORE, REF)
            VALUES (?, ?, ?, ?, ?, ?)
            '''
            cursor.execute(insert_query, (cve_id, cwe_id, title,
                                           language, cvss_score, references))
            conn.commit()

```

```
# 查询CVE信息
```

```
def get_cve_info(cve_id):  
    select_query = '''  
    SELECT * FROM CVE_INFO WHERE CVE_ID=?  
    '''  
    result = cursor.execute(select_query, (cve_id,))  
    return result.fetchone()
```

```
# 修改CVE信息
```

```
def update_cve_info(cve_id, cwe_id, title, language, cvss_score, references):  
    update_query = '''  
    UPDATE CVE_INFO  
    SET CWE_ID=?, TITLE=?, LANGUAGE=?, SCORE=?, REF=?  
    WHERE CVE_ID=?  
    '''  
    cursor.execute(update_query, (cwe_id, title, language,  
                                   cvss_score, references, cve_id))  
    conn.commit()
```

```
# 示例使用：插入CVE信息
```

```
# cve_id = 'CVE-2023-30213'  
# cwe_id = 'CWE-79'  
# title = 'WordPress Sticky Ad Bar Plugin <= 1.3.1 is vulnerable to Cross Site S  
# language = 'PHP'  
# cvss_score = 8.2  
# references = 'https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-30213'
```

```
base_path = r'cvelistV5\cves'  
for i in range(2023, 2024):  
    folder_path = os.path.join(base_path, str(i))  
    for root, dirs, files in os.walk(folder_path):  
        # THRESHOLD = 100  
        for file in files:  
            # if THRESHOLD == 0:  
            #     break  
            THRESHOLD -= 1  
            file_path = os.path.join(root, file)  
            # print(file_path)  
            cve_id, cwe_id, title, language, cvss_score, references = get_cve_fr  
                file_path)  
            try:  
                insert_cve_info(cve_id, cwe_id, title, language,  
                                cvss_score, references)  
            except:  
                pass
```

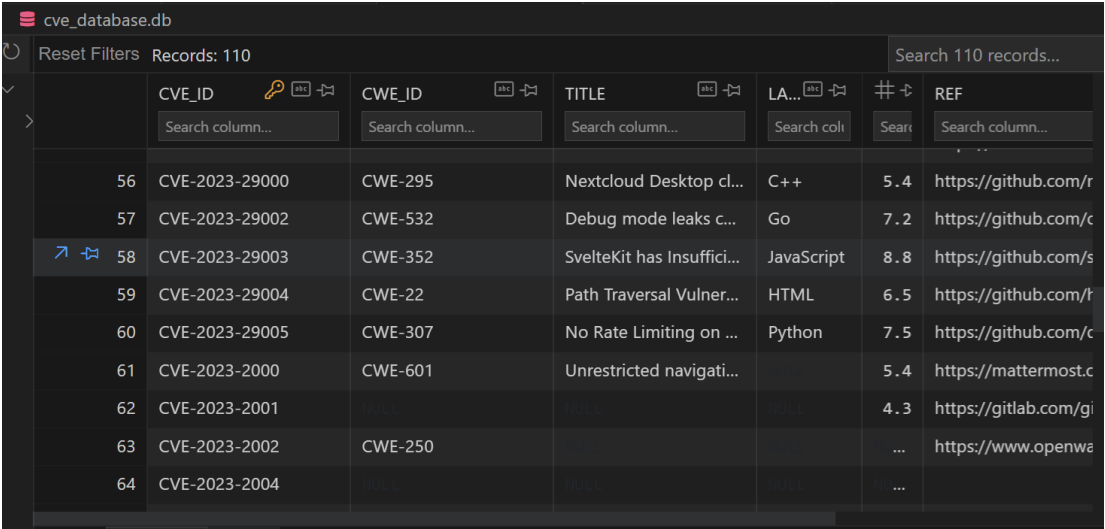
```
print('CVE information inserted.')
```

```
# 示例使用：查询CVE信息
```

```
result = get_cve_info(cve_id)  
conn.close()
```

CVE information inserted.

运行结果示例

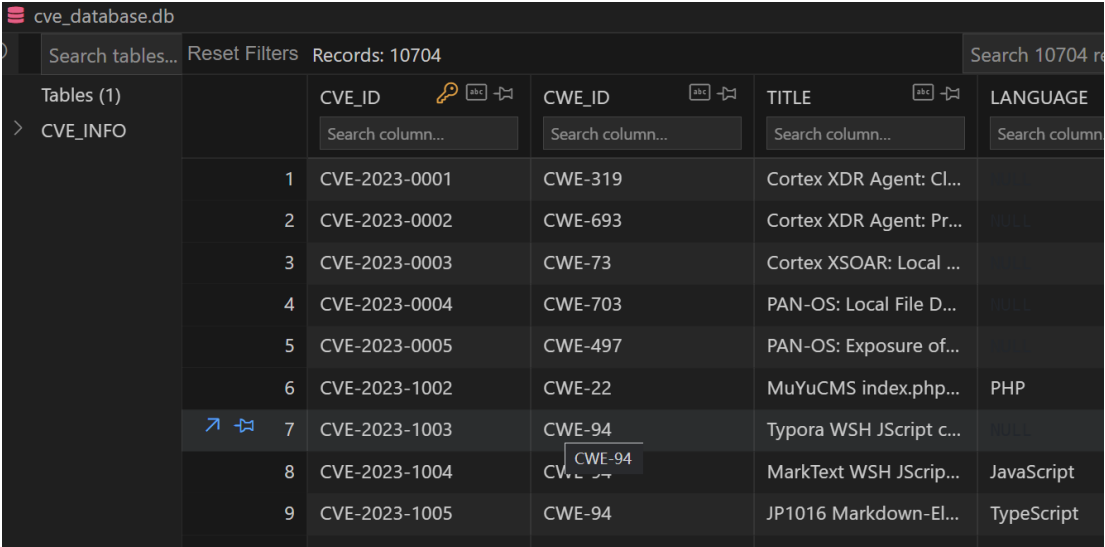


	CVE_ID	CWE_ID	TITLE	LA...	#	REF
56	CVE-2023-29000	CWE-295	Nextcloud Desktop cl...	C++	5.4	https://github.com/r
57	CVE-2023-29002	CWE-532	Debug mode leaks c...	Go	7.2	https://github.com/c
58	CVE-2023-29003	CWE-352	SvelteKit has Insuffici...	JavaScript	8.8	https://github.com/s
59	CVE-2023-29004	CWE-22	Path Traversal Vulner...	HTML	6.5	https://github.com/r
60	CVE-2023-29005	CWE-307	No Rate Limiting on ...	Python	7.5	https://github.com/c
61	CVE-2023-2000	CWE-601	Unrestricted navigati...		5.4	https://mattermost.c
62	CVE-2023-2001				4.3	https://gitlab.com/gi
63	CVE-2023-2002	CWE-250			...	https://www.openwa
64	CVE-2023-2004				...	

成功提取到了 2023 年的所有 CVE 漏洞要求的相关信息+漏洞标题。

但由于 GitHub API 的限制, 访问受限, 所以最终结果只有一部分 CVE 信息包含漏洞源代码信息。

最终记录 10704 条



	CVE_ID	CWE_ID	TITLE	LANGUAGE
1	CVE-2023-0001	CWE-319	Cortex XDR Agent: Cl...	
2	CVE-2023-0002	CWE-693	Cortex XDR Agent: Pr...	
3	CVE-2023-0003	CWE-73	Cortex XSOAR: Local ...	
4	CVE-2023-0004	CWE-703	PAN-OS: Local File D...	
5	CVE-2023-0005	CWE-497	PAN-OS: Exposure of...	
6	CVE-2023-1002	CWE-22	MuYuCMS index.php...	PHP
7	CVE-2023-1003	CWE-94	Typora WSH JScript c...	NULL
8	CVE-2023-1004	CWE-94	MarkText WSH Jscrip...	JavaScript
9	CVE-2023-1005	CWE-94	JP1016 Markdown-El...	TypeScript

总结与改进

目前来讲, 代码的主要问题如下:

- 鲁棒性较差, 对一些特殊情况的考虑覆盖不够全面, 未能全面的覆盖所有可能的字段位置, 即 JSON 文件是不规则的, 对 GitHub API 调用的考虑情况也不够周全, 存在 API 访问限制等情况.
- 速度较慢, 需要调用在线 API 处理, 且大量小文件维持读操作, 导致程序所消耗时间较长, 仅 2023 年一年 CVE 信息的处理就消耗了 1 分 40 秒+.

3. 代码工程化程度不够高, 目前代码还是处在 Demo 状态, 仅能保证在给定的文件范围内运行不出错误, 在实际应用中还需要进一步的优化与测试, 例如将代码封装成可调用的对象等.