漏洞数据库设计

数据

编号	名称	样例	类型
1	CVE ID	CVE-1999-0001（CVE+年份+4或5位随机数字)	CHAR
2	CVE ASSIGNER	cve@mitre.org	CHAR
3	vendor	freebsd	CHAR
4	product	freebsd	CHAR
5	affected versions	字符串数组，里面有多条数据	外键约束
6	CVSS (version 3.x) score	null	
7	CVSS (version 2.x) score	5.0	
8	CWE IDs	数组，里面有多条字符串数据	
9	publishedDate	1999-12-30	DATE
10	lastModifiedDate	2010-12-16	DATE
11	CVE description	字符串	CHAR
12	references	字符串数组	
13	patches	字符串数组	
14	fix	字符串	CHAR
15	CWE	字符串漏洞类型	CHAR
	以上为json中的内容	以下为尚未拿到的数据内容	
16	引用次数	数据库设计//认为是查找次数	INT
17	代码片段	字符串数组	
18	代码位置	字符串数组	
19	漏洞成因	字符串	
20	漏洞处置建议	字符串	
21	漏洞来源平台		CHAR
22	漏洞来源平台链接	字符串	
23	野外漏洞报告	字符串	

编号	名称	样例	类型
24	漏洞PoC	字符串	
25	漏洞危害	字符串	
26	触发性评分		FLOAT
	会议中未提到	但是好像需要用到的数据	
27	漏洞修复率	把是否有fix版本设置为漏洞修复率	FLOAT
28	每个漏洞发现的时间		DATE

需求

编号	需求	实现方式
1	漏洞类型	根据漏洞的编号查找漏洞的类型CWE
2	漏洞修复率	根据漏洞的编号查找漏洞的类型
3	数据库中漏洞的查询引用次数前30	编号-查询次数，根据查询次数进行排序
4	新增漏洞个数	根据漏洞时间把符合条件的漏洞统计出数目
5	月度更新的最新漏洞实例	暂无法实现
6	组件+版本得到漏洞的数量	vender+product 对应的漏洞的数量
7	影响组件对应的代码片段	根据CVE ID 查询到代码片段，把代码片段全部输出
8	影响组件对应的代码位置	根据CVE ID 查询到代码位置，把代码位置全部输出
9	漏洞触发条件	可以用漏洞成因来代替?

设计

全文索引 > 联合索引 > json函数查询 > like查询

故放弃使用mysql中json数据类型进行存储

思路

以CVE ID为主索引，原始数据中以字符串或数值形式存储的数据直接存储在主表中，将原始数据中以数组形式存储的数据转化为以CVEID为主键和外键的子表中

CVE ID	CVESSIGNER	vendor	product	CVSS_score_3	CVSS_score_2	publishedDate	lastModifiedDate	CVEDescription	fix	counts

CVE ID	version

CVE ID	CWE ID

CVE ID	Reference

CVE ID	Patch

CVE ID	CVE_Description
	注：为中文翻译版

建表

```
create database Vulnerability;
use vulnerability

CREATE TABLE CVE(
    CVE_ID CHAR(15) PRIMARY KEY NOT NULL,
    CVESSIGNER VARCHAR(500),
    vendor VARCHAR(500),
    product VARCHAR(500),
    CVSS_score_3 FLOAT,
    CVSS_score_2 FLOAT,
    publishedDate DATE,
    lastModifiedDate DATE,
    CVEDescription VARCHAR(1000),
    fix VARCHAR(500),
    counts INT
);



CREATE TABLE Versions (
    CVEID CHAR(15) ,
    version VARCHAR(500)
);


CREATE TABLE CWEIDS (
    CVEID CHAR(15) ,
    CWEID CHAR(20)
```

```
);


CREATE TABLE Reference (
    CVEID CHAR(15) ,
    reference VARCHAR(500)
);


CREATE TABLE Patches (
    CVEID CHAR(15),
    patch VARCHAR(255)
);

CREATE TABLE CVE_DES (
    CVEID CHAR(15),
    CVEDes VARCHAR(500)
);

```

载入数据

示例

3||张三||22||北京||2012-09-19 00:00:00
4||李明||32||\N||2017-05-12 00:00:00
5||孙权||12||广州||\N

可以直接将存储这种数据的.txt文件使用Load函数批量的将数据导入

写了一个简单的python将已有的json数据转换成这种格式

例:

CVE-1999-0001||cve@mitre.org||freebsd||freebsd||NULL||NULL||1999-12-30||2010-12-16||ip_input.c in BSD-derived TCP/IP implementations allows remote attackers to cause a denial of service (crash or hang) via crafted packets.||NULL

```
load data local infile "D:\\2023Spring\\漏洞数据库\\数据库\\CVE.txt" into table cve
CHARACTER SET utf8
FIELDS TERMINATED BY '     '
OPTIONALLY ENCLOSED BY ''
LINES TERMINATED BY '\n'
(CVE_ID,CVESSIGNER,vendor,product,CVSS_score_3,CVSS_score_2,publishedDate,lastModifiedDate,CVEDescription,fix,counts);

load data local infile "D:\\2023Spring\\漏洞数据库\\数据库\\Patches.txt" into table patches
CHARACTER SET utf8
FIELDS TERMINATED BY '||'
 OPTIONALLY ENCLOSED BY ''
LINES TERMINATED BY '\n'
(CVEID,patch) ;

```

```
load data local infile "D:\\2023Spring\\漏洞数据库\\数据库\\Reference.txt" into table Reference
CHARACTER SET utf8
FIELDS TERMINATED BY '||'
 OPTIONALLY ENCLOSED BY ''
LINES TERMINATED BY '\n'
(CVEID,reference) ;

load data local infile "D:\\2023Spring\\漏洞数据库\\数据库\\Versions.txt" into table Versions
CHARACTER SET utf8
FIELDS TERMINATED BY '||'
 OPTIONALLY ENCLOSED BY ''
LINES TERMINATED BY '\n'
(CVEID,version) ;

load data local infile "D:\\2023Spring\\漏洞数据库\\数据库\\CWEIDs.txt" into table CWEIDS
CHARACTER SET utf8
FIELDS TERMINATED BY '||'
OPTIONALLY ENCLOSED BY ''
LINES TERMINATED BY '\n'
(CVEID,CWEID) ;
-- 注：文件地址需自行更改

load data local infile "D:\\2023Spring\\供应链项目\\漏洞数据库\\数据库\\漏洞数据库\\CVE Des_CN.txt" into table CVE_DES
CHARACTER SET utf8
FIELDS TERMINATED BY '||'
OPTIONALLY ENCLOSED BY ''
LINES TERMINATED BY '\n'
(CVEID,CVEDes) ;
```

注：在本地实现，可使用上传的几个.txt文件

查询语句

```
-- 1.查询漏洞类型 id
-- 实现方式：根据CVE ID查找存储的CWEIDS
-- 这里注意输入的id需要加引号，构成字符串
select CWE from cweids where CVE_ID=id;
update cve set counts=counts+1 where CVE_ID=id;

-- 2.漏洞修复率 id
select fix from cve where CVE_ID=id;
update cve set counts=counts+1 where CVE_ID=id;

-- 3.数据库中漏洞的查询引用次数前30
SELECT * FROM cve ORDER BY counts LIMIT 0,30;

-- 4.新增漏洞个数（暂无数据）

-- 5.月度更新的最新漏洞实例（暂无数据）
```

```
17
18 -- 6.组件+版本得到漏洞的数量 v (版本中已有组件信息，实际查询时只需version)
19 select * from versions where version = v;
20 -- select * from versions where version =
   'cpe:2.3:o:sgi:irix:5.3:*:*:*:*:*:*:*';
21
22 -- 以下内容数据库中暂无数据
23 -- 7.影响组件对应的代码片段
24
25 -- 8. 影响组件对应的代码位置
26
27 -- 9. 漏洞触发条件
28
29
```

mysql常用语句

```
1  -- 登录
2  mysql -u root -p
3
4  -- 查看有哪些数据库
5  show databases;
6
7  -- 使用某个数据库
8  use vulnerability
9
10 -- 查看数据库中的数据表
11 show tables;
12
13 -- 查看表的结构
14 describe tablename;
15 desc tablename;
16
17 -- 删除某个表
18 drop table tablename
19
20 -- 删除某张表中的所有数据
21 delete from tablename;
22 truncate table tablename;
23
24 -- 更改字段长度
25 alter table tablename modify column columnname 类型(要修改的长度);
26 alter table Patches modify column CVEID char(14);
27 alter table Patches modify column patch char(255); -- 255 为CHAR的最大长度
28
29 -- 查找某个组件的版本等信息
30 SELECT * FROM versions WHERE version LIKE "%:spark:%";
31
32 -- 批量查找数据
33 select CVE_ID, CVSS_score_3, CVSS_score_3 from cve where CVE_ID in ("CVE-
   2017-100049", "    ");
```

报错解决

```
-- 1. Error 2068
-- 以mysql -u 用户名 -p --local-infile的命令登陆
mysql -u root -p --local-infile

-- 2. ERROR 3948 (42000): Loading local data is disabled; this must be enabled on both the client and server sides
set global local_infile=on;
```

简单代码

为方便查询，简单的写了一个代码进行语句的转换

输入文件的内容为：CVE编号（在.txt文件中可有多个）

输出内容为查询语句

```
#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <sstream>
#include <tchar.h>

using namespace std;

int main()
{

    string s;
    ifstream fin;
    fin.open("D:\\2023Spring\\C++Code\\select_data\\data.txt");
    if (fin.is_open())
    {
        ofstream out;
        out.open("D:\\2023Spring\\C++Code\\select_data\\answer1.txt");
        out << "select CVE_ID, CVSS_score_3, CVSS_score_3 from cve where CVE_ID in (";
        while (getline(fin, s)){
            s.erase(s.find_last_not_of(" ") + 1);
            out << "\"" << s << "\""
                << ",";
        }
        out << ");";
    }
}
```

####