

漏洞数据库设计

数据

编号	名称	样例	类型	按需是否被使用到
1	CVE ID	CVE-1999-0001 (CVE+年份+4或5位随机数字)	CHAR	是 ok
2	CVE ASSIGNER	cve@mitre.org	CHAR	
3	vendor	freebsd	CHAR	是 ok
4	product	freebsd	CHAR	是ok
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		ok
9	publishedDate	1999-12-30	DATE	
10	lastModifiedDate	2010-12-16	DATE	
11	CVE description	字符串	VARCHAR	
12	references	字符串数组		
13	patches	字符串数组		
14	fix	字符串	CHAR	ok
15	CWE	字符串漏洞类型	CHAR	这个应该和CWE-ID重复了 ok
	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	是 ok
17	代码片段	字符串数组		是 ok
18	代码位置	字符串数组		是 ok
19	漏洞成因	字符串		是? ok
20	漏洞处置建议	字符串		

编号	名称	样例	类型	按需是否被使用到
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		
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	是
28	每个漏洞发现的时间		DATE	是
29				

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	?
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?

编号	需求	实现方式
	以下为未提到的需求	
10	未有明确的需求一并设置为根据ID	
	进行查询的简单模式	

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	CVESIGNER	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

建表

```
1  create database vulnerability;
2  use vulnerability
3
4  CREATE TABLE CVE(
5      CVE_ID CHAR(14) PRIMARY KEY NOT NULL,
6      CVESSIGNER VARCHAR(500),
7      vendor VARCHAR(500),
8      product VARCHAR(500),
9      CVSS_score_3 FLOAT,
10     CVSS_score_2 FLOAT,
11     publishedDate DATE,
12     lastModifiedDate DATE,
13     CVEDescription VARCHAR(1000),
14     fix VARCHAR(500),
15     counts INT
16 );
17
18     --CodeSegment VARCHAR(1000),
19     --CodeLocation VARCHAR(500),
20     --cause VARCHAR(500),
21     --suggestion VARCHAR(500),
22     --SourcePlatform VARCHAR(500),
23     --SourcePlatformLink VARCHAR(500),
24     --Reportsinwild VARCHAR(500),
25     --PoC VARCHAR(500),
26     --damages VARCHAR(500)
27
28
29  CREATE TABLE Versions (
30      CVEID CHAR(14) PRIMARY KEY,
31      version VARCHAR(500),
32      FOREIGN KEY(CVEID) REFERENCES cve(CVE_ID)
33 );
34
35
36  CREATE TABLE CWEIDS (
37      CVEID VARCHAR(14) PRIMARY KEY,
38      CWEID VARCHAR(500) ,
39      FOREIGN KEY(CWEID) REFERENCES cve(CVE_ID)
40 );
41
42
43  CREATE TABLE Reference (
44      CVEID CHAR(14) PRIMARY KEY,
45      reference VARCHAR(500),
46      FOREIGN KEY(CVEID) REFERENCES cve(CVE_ID)
47 );
48
49
50  CREATE TABLE Patches (
51      CVEID CHAR(10),
52      patch CHAR(30)
53 );
```

54

55 ALTER TABLE cve alter column counts set default 0;

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

```
1 load data local infile "D:\\漏洞数组库\\SubtableList\\CVE.txt" into table cve
2 CHARACTER SET utf8
3 FIELDS TERMINATED BY '||'
4 OPTIONALLY ENCLOSED BY ''
5 LINES TERMINATED BY '\n'
6 (CVE_ID,CVESSIGNER,vendor,product,CVSS_score_3,CVSS_score_2,publishedDate,lastModifiedDate,CVEDescription,fix,counts);
7
8 load data local infile "D:\\漏洞数组库\\SubtableList\\Patches.txt" into table patches
9 CHARACTER SET utf8
10 FIELDS TERMINATED BY '||'
11 OPTIONALLY ENCLOSED BY ''
12 LINES TERMINATED BY '\n'
13 (CVEID,patch) ;
14
15 load data local infile "D:\\漏洞数组库\\SubtableList\\CWEIDs.txt" into table Reference
16 CHARACTER SET utf8
17 FIELDS TERMINATED BY '||'
18 OPTIONALLY ENCLOSED BY ''
19 LINES TERMINATED BY '\n'
20 (CVEID,reference) ;
21
22 load data local infile "D:\\漏洞数组库\\SubtableList\\Versions.txt" into table Versions
23 CHARACTER SET utf8
24 FIELDS TERMINATED BY '||'
25 OPTIONALLY ENCLOSED BY ''
26 LINES TERMINATED BY '\n'
27 (CVEID,version) ;
28
29 load data local infile "D:\\漏洞数组库\\SubtableList\\CWEIDs.txt" into table CWEIDS
30 CHARACTER SET utf8
```

```
31 FIELDS TERMINATED BY '||'
32 OPTIONALLY ENCLOSED BY ''
33 LINES TERMINATED BY '\n'
34 (CVEID,CWEID) ;
```

注：在本地实现，可使用上传的几个.txt文件

查询语句

```
1  -- 1.查询漏洞类型 id
2  -- 实现方式：根据CVE ID查找存储的CWEIDS
3  -- 这里注意输入的id需要加引号，构成字符串
4  select CWE from cve where CVE_ID=id;
5  update cve set counts=counts+1 where CVE_ID=id;
6
7  -- 2.漏洞修复率 id
8  select fix from cve where CVE_ID=id;
9  update cve set counts=counts+1 where CVE_ID=id;
10
11 -- 3.数据库中漏洞的查询引用次数前30
12 SELECT * FROM cve ORDER BY counts LIMIT 0,30;
13
14 -- 4.新增漏洞个数
15
16 -- 5.月度更新的最新漏洞实例
17
18 -- 6.组件+版本得到漏洞的数量 v
19 select * from versions where version = v;
20 -- select * from versions where version =
21 'cpe:2.3:o:sgi:irix:5.3:*:*:*:*:*:*';
22
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
```

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
```

报错解决

```
1 -- 1. Error 2068
2 -- 以mysql -u 用户名 -p --local-infile的命令登陆
3 mysql -u root -p --local-infile
4
5 -- 2. ERROR 3948 (42000): Loading local data is disabled; this must be
   enabled on both the client and server sides
6 set global set local_infile=on;
```