PostgreSQL 高权限命令执行 (CVE-2019-9193)漏洞复现&实战引申

0x01 漏洞利用

这个漏洞是一个版本漏洞

从9.3版本开始，Postgres新增了一个 `COPY TO/FROM PROGRAM`功能，允许数据库的超级用户以及 `pg_read_server_files`组中的任何用户执行操作系统命令，利用的前提

1、需要登录 2、需要高权限

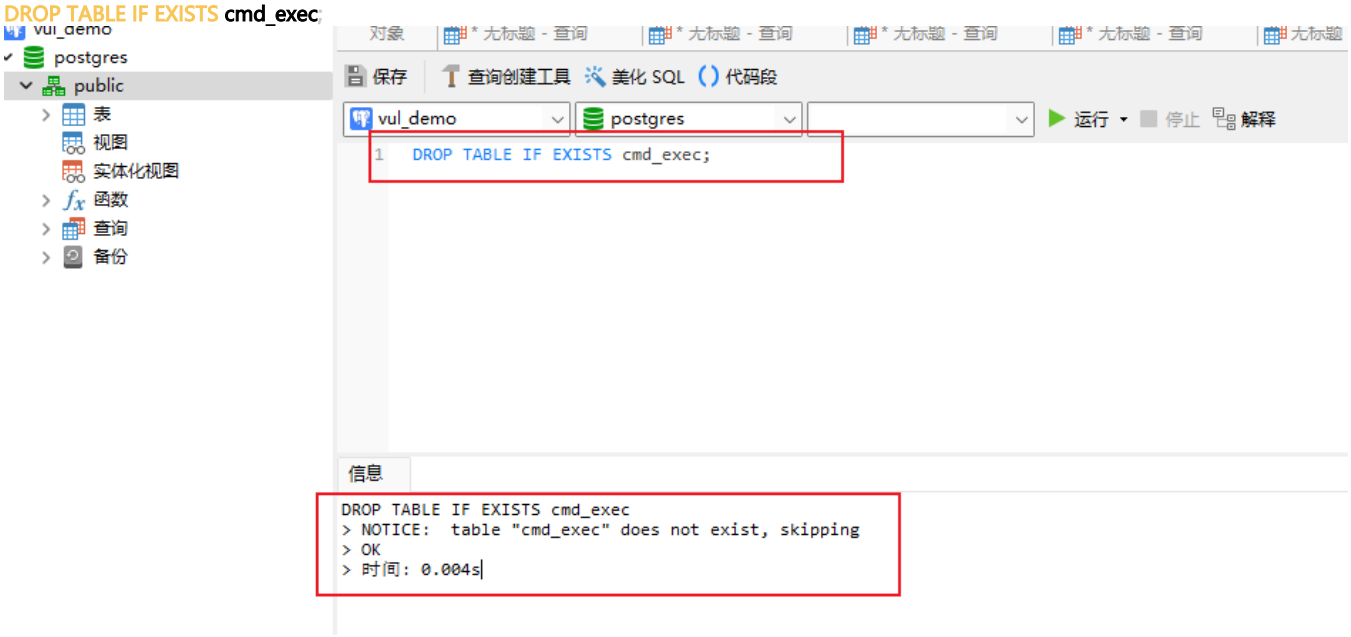
所以要先弱口令爆破之后，然后查看是否是高权限，这个漏洞才有得玩。

这里之所以选择复现这个漏洞，因为其中的思路比较有意思，也就是看回显的思路。

先大体讲一下这个漏洞是怎么看回显的，他是通过命令执行然后把回显写入创建的一张表里面，下面来具体操作。

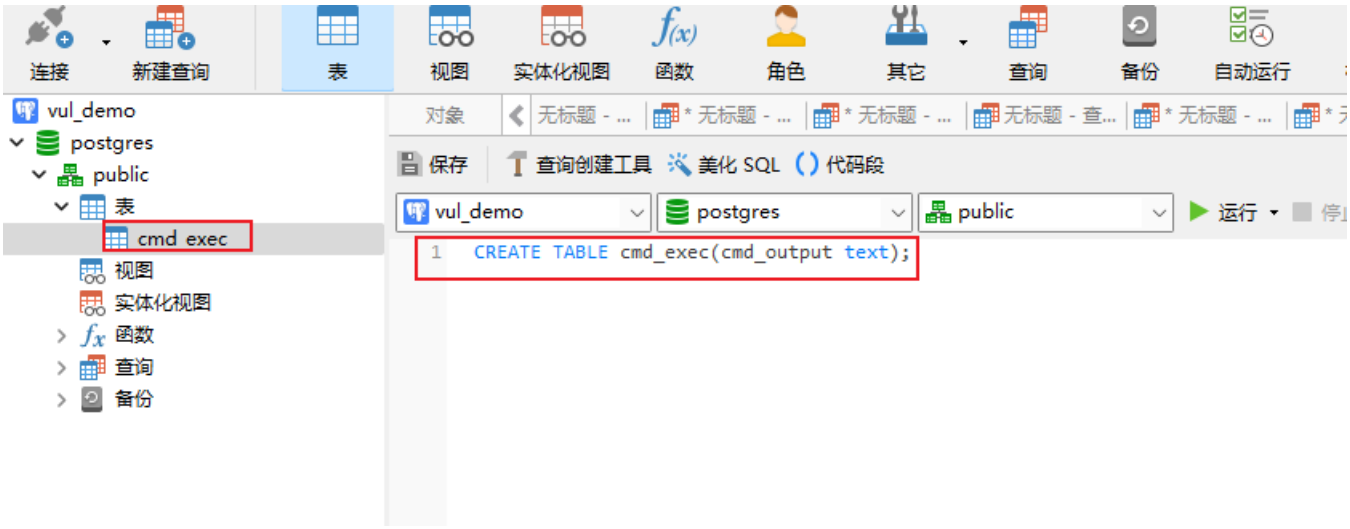
- **先删除你想要使用但是已经存在的表**

```
DROP TABLE IF EXISTS cmd_exec;
```



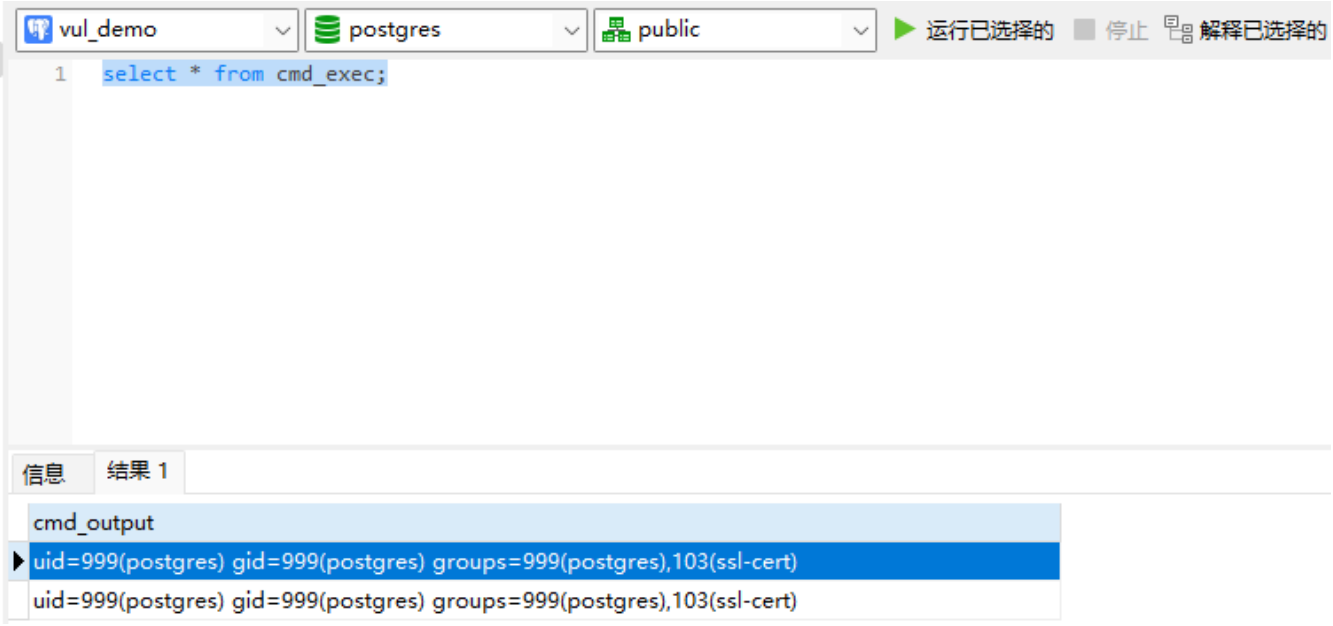
- **创建保存系统命令输出的表**

```
CREATE TABLE cmd_exec(cmd_output text);
```



- **执行系统命令 利用特定函数 copy xxxx from program**

查看执行结果



这里要强调的点就是命令执行的看回显方法，其他倒是没什么，我觉得这个写入表来看回显的方法很好玩。

0x02 实战引申

那么下面给一个实战的例子 也是sql注入没有回显 然后通过把命令执行的结果写入表中 然后用select看回显 由于项目敏感这里只给出部分截图和思路

这里的条件 1、sql注入没有回显 2、后台是mssql可以执行xp_cmdshell 3、后台有功能可以select查看对应的表

- **通过抓取后台和数据库相关的接口找到这个接口**

/PH_SystemModule/DatabaseTable/GetList?tableName=test&databaseLinkId=f2d587de-43e5-4310-b968-4544f4961a39&_=1634809489388

- **通过对应接口审计源码**

上述接口中获取到的关键词为

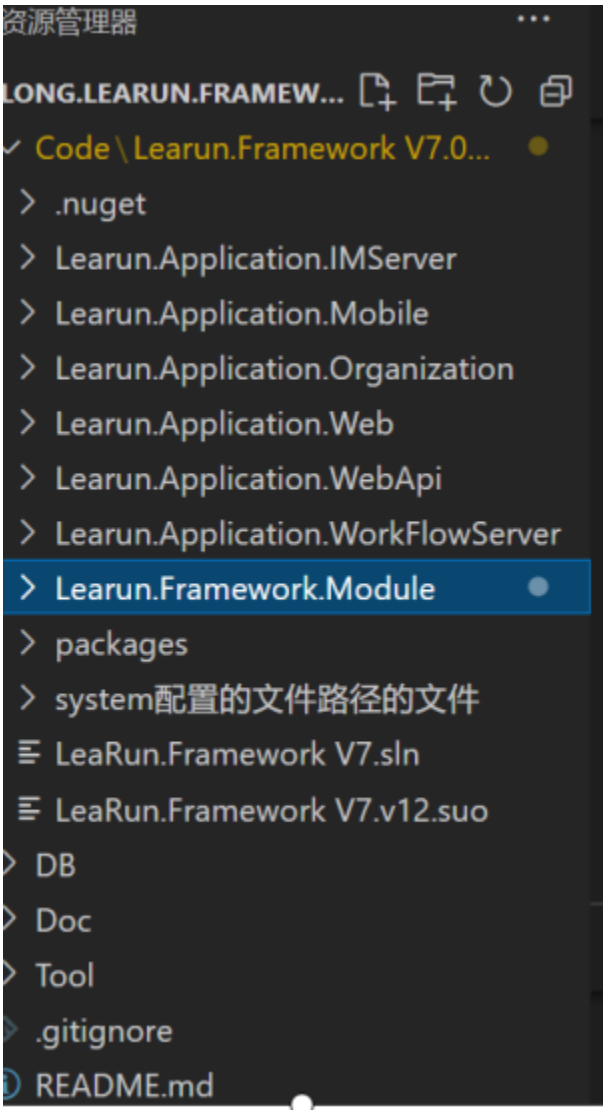
PH_SystemModule

DatabaseTable

GetList

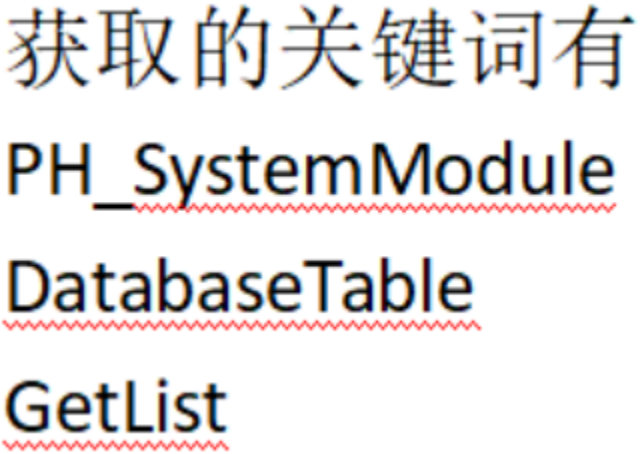
应该是目录结构 先记下来

然后看到项目的路径

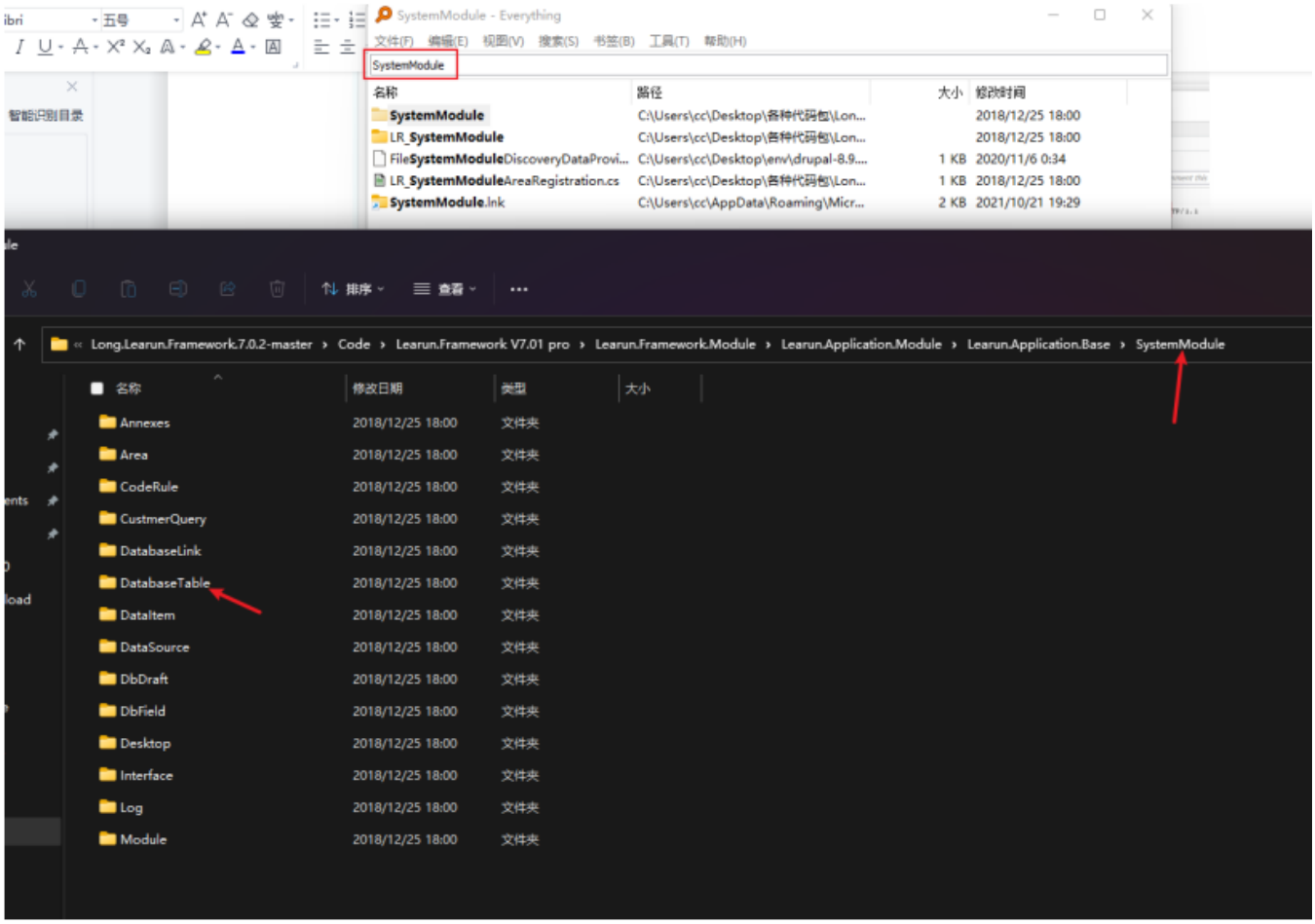


是一个比较典型的mvc架构的项目，虽然不是熟悉的语言，但是架构都是共通的，其中module代表数据库相关的代码

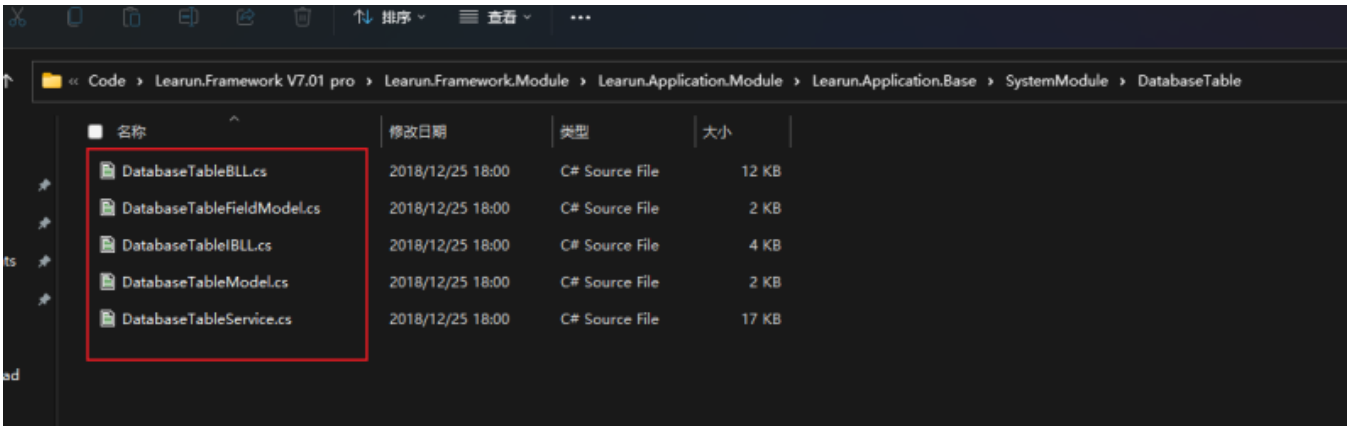
再结合之前的路径名



- **everything全局搜索对应的关键词**



- **成功定位到对应的代码文件**



- **打开代码进行审计**

这个strSql是直接从前台传入的，用?strSql=传入即可

```
public List<string> GetSqlColName(DatabaseLinkEntity databaseLinkEntity, string strSql)
{
    try
    {
        var dt = this.BaseRepository(databaseLinkEntity.F_DbConnection, databaseLinkEntity.F_DbType).FindTable(strSql);
        List<string> res = new List<string>();
        foreach (DataColumn item in dt.Columns)
        {
            res.Add(item.ColumnName);
        }
        return res;
    }
    catch (Exception ex)
    {
        if (ex is ExceptionEx)
        {
            throw;
        }
        else
```

跟进FindTable方法看看有没有waf

```
10 references
public DataTable FindTable(string databaseLinkId, string sql, object dbParameter = null)
{
    try
    {
        DatabaseLinkEntity entity = GetEntity(databaseLinkId);
        return databaseLinkService.FindTable(entity, sql, dbParameter);
    }
    catch (Exception ex)
    {
        if (ex is ExceptionEx)
        {
            throw;
        }
        else
        {
```

```
private DatabaseLinkService databaseLinkService = new DatabaseLinkService();
#endregion

#region 缓存定义
8 references
private ICache cache = CacheFactory.CaChe();
8 references
private string cacheKey = "learun_adms_databaseLink";
#endregion
```

没有发现waf，可以直接构造参数来进行注入

然后通过这个注入点注入进去之后，后台并没有回显，于是构造语句写入数据库的表中然后后台用select进行查看

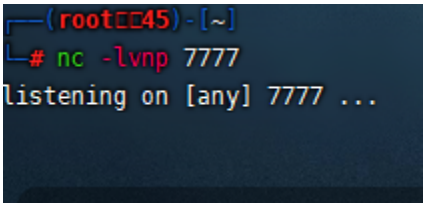
这里不能放图了，只能描述了，大体就是这么个思路，用来看命令执行的回显。

0x03反弹shell

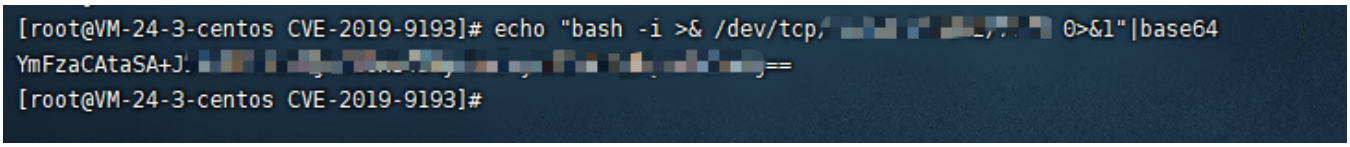
- **这里的漏洞其实可以使用反弹shell，但是直接执行会有问题，看例子**

直接报错了

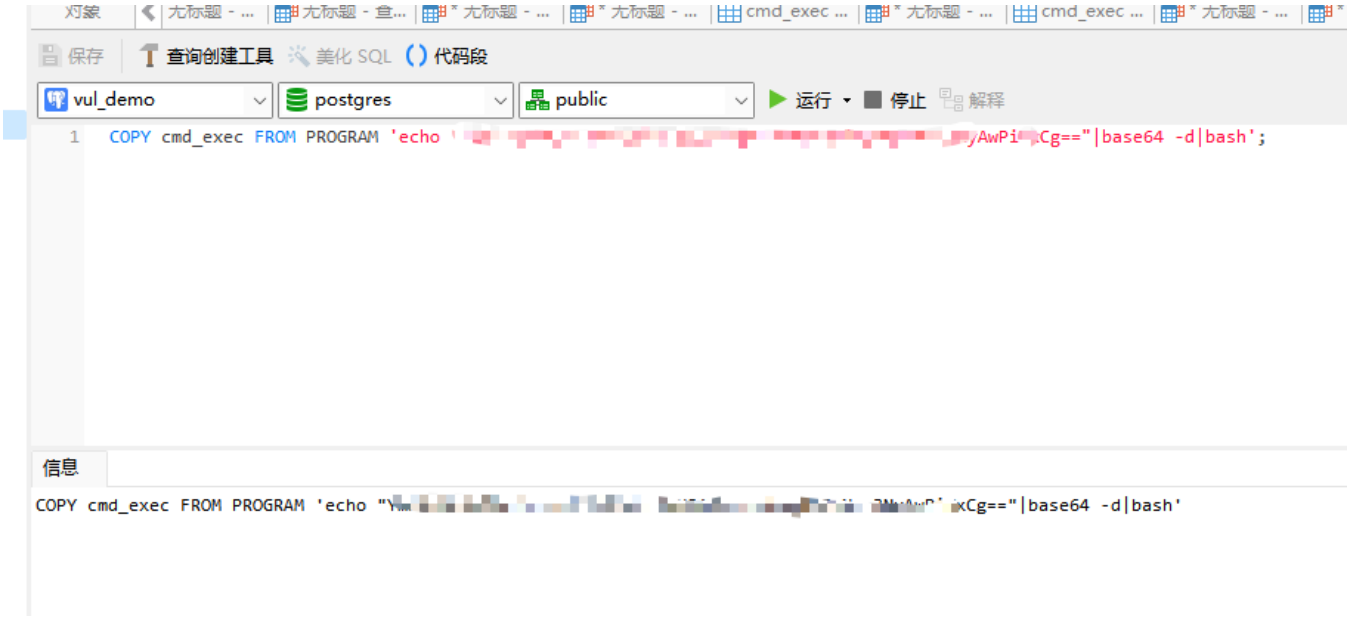
- **vps也没有得到shell**

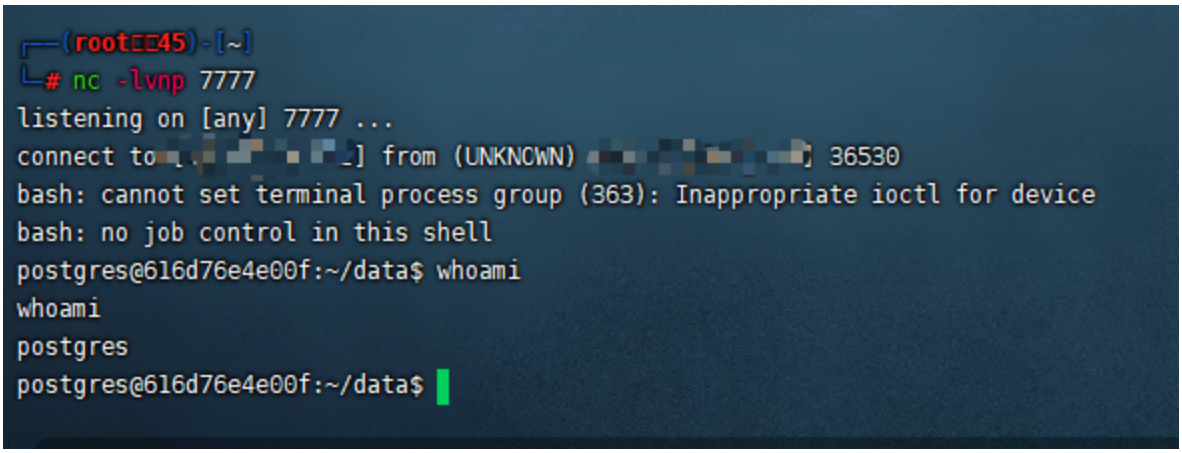


这里可以用编码来解决



- **执行命令，直接上线**





这里的编码不仅仅限于base64 b32 等等都可以 主要是为了解决数据传输过程中的特殊字符被异常解析的问题