HZ BOOKS
华章科技

· 网络空间安全技术丛书 ·

ANDROID APPLICATION SECURITY PROTECTING AND REVERSE ANALYSING

Android应用安全防护和逆向分析

姜维 编著

- 360公司创始人周鸿祎、CSDN创始人蒋涛、看雪学院创始人段钢等联袂推荐。
- 零基础学习移动安全技术，手把手带你进入逆向分析领域。

机械工业出版社
China Machine Press

· 网络空间安全技术丛书 ·

Android应用安全防护和逆向分析

ANDROID APPLICATION SECURITY PROTECTING AND REVERSE ANALYSING

姜维 编著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Android 应用安全防护和逆向分析 / 姜维编著 . —北京：机械工业出版社，2017.11
（网络空间安全技术丛书）

ISBN 978-7-111-58445-2

I. A… II. 姜… III. 移动终端 - 应用程序 - 程序设计 - 安全技术 IV. TN929.53

中国版本图书馆 CIP 数据核字（2017）第 274118 号

Android 应用安全防护和逆向分析

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：吴 怡
印 刷：北京诚信伟业印刷有限公司
开 本：186mm × 240mm 1/16
书 号：ISBN 978-7-111-58445-2
版 次：2018 年 1 月第 1 版第 1 次印刷
印 张：27.25（含 0.5 印张彩插）
定 价：99.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426 88361066
购书热线：（010）68326294 88379649 68995259
投稿热线：（010）88379604
读者信箱：hzit@hzbook.com

对本书的赞誉

移动应用安全已经是个不可忽视的问题，本书结合多个案例进行实际操作讲解，介绍很多实用工具和命令，是一本非常好的安全逆向工具书籍。姜维研究安全逆向这么久，将自身多年的丰富经验用书本的形式展现给读者也是非常不易。我仔细看过这本书，感觉真的很不错，对学习安全逆向的同学很有帮助，值得购买。

——周鸿祎，360 公司创始人、董事长兼 CEO，知名天使投资人

本书作者的技术搏客在 CSDN 上有超过 500 万人次的总访问量！

为什么这么多人会关注 Android 安全技术？因为不仅 Android 安全技术越来越重要，而且要成为真正的开发高手需要从正反两个方面来学习：从正面学习 Android 编程类别 API，逐层学习 App 开发，从反面（逆向研究 APK 代码）学习 Android 系统如何运行的，其他软件是如何实现特定功能的，这样获得的 Android 知识更加系统全面。作者在书中有非常详细的案例讲解，也提供了大量的工具源码，是 Android 开发人员逆向学习研究的极好工具手册，可以帮助 Android 开发者成为更全面的 Android 高手。

——蒋涛，CSDN 创始人，极客帮基金创始人

Android 系统应用面越来越广，从移动手机到智能硬件，都有很高的覆盖率，其引发的安全问题危害也越来越大。作者以实际操作讲解的同时，还特别重视一些原理的介绍，可以帮助读者快速提升 Android 安全技术水平。

——段钢，看雪学院创始人

安全领域是一个冒险的世界，作者结合自己在这个领域的丰富经验，用生动的案例带领读者在这个世界中探险。本书章节安排精心合理，技术点循序渐进，使用了大量带有标注的配图，清晰且直观，属于难得一见的用心之作。

——泮晓波，顶象技术移动安全实验室负责人、合伙人

本书的作者在信息安全领域的多年经验，加上纯粹的技术分享精神，注定了这是一本值得广大安全爱好者期待的好书，本书完整地向你诠释——掌握了安全技术的底层原理，你就掌握了安全的一切。

——丰生强（网名：非虫），国内知名安全专家

随着移动应用的普及，移动应用的安全也随之成为一个严肃而重要的问题。要解决该问题所需要学习的内容非常多，对初学者而言，一本合适的入门书籍将起到承前启后的重要作用。而本书就恰恰是这样的一本书籍，它既涵盖了入门者所必须掌握的基础知识，又精选了许多安全领域中的典型实例并对它们开展了详细讲解。最后，希望作者能再接再厉，对本书内容进行持续更新。

——邓凡平

前　言

随着移动应用的广泛使用，不可忽视的一个问题就是信息安全。本书从 Android 应用安全和逆向两个方面来介绍移动开发中如何做好安全。

我本来不是从事安全逆向工作的，但是一个偶然的机会使我发现逆向研究非常有趣，因为在逆向出别人的 App 那一刻会觉得无比自豪。

第一次逆向是因为遇到一个问题需要去解决，记得当时想查看一个 App 的内部资源信息，尝试使用 apktool 反编译程序，惊奇地发现原来别人的 App 反编译之后会有这么多东西，逆向真的很有趣。而后在开发中无法实现或者没有相应的资源时就去反编译别人的 App，查看对应的代码怎么实现。后来就反编译那些有阻碍的应用，慢慢摸索，克服困难，每研究成功一个案例，便是对自身技能的一次提高。从第一次使用 apktool 工具，到使用 Jadx 可视化工具，再到用 IDA 工具调试，一步一个脚印走过来，经验逐渐丰富，技能也慢慢提升了。

逆向研究需要一种逆向思维，我没有接受过专业训练，只是在业余时间看相关书籍，找几个样本研究，从简单到难，在这个过程中慢慢学到很多技术。本书就是我这几年学习与探索的总结。

本书涉及内容有点多，但是没有一章是多余的，每章内容都是干货。本书包括 26 章，分为四篇。

基础篇

基础篇包括第 1 ～ 7 章，主要介绍 Android 技术中与逆向相关的基础知识，为后续章节的学习做准备。

第 1 章通过对 Android 中锁屏密码加密算法的分析，带领读者进入安全世界，这方面内容不算复杂，但是需要阅读 Android 源码来得到算法分析，其中一个知识点是如何通过查看 Android 源码来帮助解决开发过程中遇到的问题，这是所有 Android 开发人员必备的技能。通过找到锁屏密码入口，一步一步跟踪最终得到密码加密算法，以及加密之后的内容存放在哪里。这是进入安全逆向分析世界的大门。

第 2 章主要介绍 Android 中 NDK 开发知识。为了安全考虑，现在很多应用把一部分功能做到了 Native 层，所以如果不知道 NDK 开发技巧，就无法进行后续的逆向操作。这一章从搭建环境到每个方法的使用，详细讲解了 Android 中 NDK 开发的技巧。

第 3 章主要介绍 Android 中开发以及逆向需要用到的命令，每个命令都有特定的案例和用法，这些命令不仅仅用于逆向，也能够帮助开发人员提升开发效率，所以了解和掌握这些命令是至关重要的。

第 4 ～ 7 章主要剖析 Android 中编译之后的 apk 包含的四类主要文件格式，这部分内容可

能有点枯燥，但是至关重要，因为在安全防护或者逆向分析中都有很重要的意义。

防护篇

防护篇包括第 8 ～ 14 章，主要介绍安全防护的相关内容，是本书的核心内容之一。

第 8 章介绍现在一些应用主要用到的安全防护策略，如混淆、签名校验、反调试检测等，每个安全策略都给出了详细介绍。

第 9 章介绍 Android 开发中经常用到的一些权限，介绍如果对这些特殊权限操作不当会带来什么样的安全问题，以及如何预防。

第 10 章介绍 Android 中的 run-as 命令以及如何分析系统安全策略，详细介绍了 App、shell、system 这三种身份，并介绍了一些技巧，比如如何对应用进行升级权限、降低权限等操作。

第 11 章讲解 Android 配置文件中的 allowBackup 属性引发的安全问题，以及如何应对。本章用一个案例来分析如何导出沙盒数据查看应用中的密码信息，修改密码信息然后再进行还原，全程无需 root 权限即可完成。

第 12 章介绍 Android 中的应用签名机制，讲解应用的签名信息是如何保存的、如何验证的，签名机制的流程，以及如何预防安全问题。

第 13 章介绍在 Android 中对 apk 进行加固的策略，以及如何对恶意者分析 apk 文件的操作进行防护，还涉及 Android 中的动态加载机制，并通过动态加载技术实现 apk 文件的解密功能。

第 14 章介绍在 Android 中如何对 so 文件加固，如何做到安全防护功能。

工具篇

工具篇包括第 15 ～ 19 章，主要介绍逆向分析需要用到的几个工具，本书从实际应用出发，详细介绍每个工具的具体用法，特别是在使用的过程中遇到问题时的处理方法。

第 15 章介绍逆向工作中用到的工具，以及如何开启设备的调试总开关，这个技能在逆向调试的时候非常重要。

第 16 章主要介绍 Android 中反编译神器 apktool 和 Jadx，详细介绍了这两个工具如何使用，以及使用过程中遇到问题时的处理方法。

第 17 章主要介绍 Android 中一款 Hook 神器 Xposed，详细介绍这个工具的用法，以及遇到问题时的处理方法。

第 18 章主要介绍一款脱壳工具，它是基于 Xposed 工具编写的，可以自动脱壳。

第 19 章主要介绍 Android 中另外一款 Hook 神器 Cydia Substrate，详细介绍了这个工具的用法以及遇到问题时的处理方法。

操作篇

操作篇包括第 20 ～ 26 章，主要介绍 Android 中的逆向操作技巧，包括静态方式和动态方式逆向，用一个经典的加固应用作为逆向案例分析了现阶段脱壳的大致流程，还介绍了 Android 开发中会遇到的系统漏洞，并分析了一个经典的病毒样本。

第 20 章主要介绍如何利用静态方式逆向应用，用一个案例讲解了静态防护逆向应用的流程。

第 21 章主要介绍如何使用 Eclipse 动态调试 smali 代码来逆向应用，用一个案例分析整个

操作流程。

第 22 章主要介绍使用 IDA 工具动态调试 so 源码，同时也介绍了一些应对反调试检测的方法。

第 23 章主要介绍如何逆向那些经过加固的应用，用一个案例详细介绍了每一步操作，最后总结了现在脱壳操作的大致流程。

第 24 章主要总结之前介绍的逆向知识，用一个经典案例作为收尾，讲解了现阶段逆向应用的大体流程和思路。

第 25 章介绍 Android 开发中会遇到的系统漏洞——一个是解压文件漏洞，一个是录屏授权漏洞。如果这两个漏洞不做修复，会导致应用沙盒数据篡改以及用户隐私数据的丢失等。该章详细介绍了漏洞的产生原因以及如何进行修复。

第 26 章介绍 Android 中一个非常经典的文件加密病毒样本，通过静态方式分析了该病毒样本的工作原理，总结了处理该类病毒的方法。

什么人适合阅读本书

阅读本书需要有一定的 Android 开发基础。有的读者可能会觉得第 1 章内容就有点深，本书第 1 章的目的在于把读者带入安全世界，看不懂没关系，可以从第 2 章开始看下去，毕竟应用开发领域和安全逆向领域有很多不一样的地方。本书最大的特点在于非常实用，用案例讲解详细操作步骤，跟着每一步具体操作，才能真正看明白。可以把本书作为一本参考书，没看懂不要急，多操作几遍试试。

欢迎联系我

本书用到了很多工具和案例样本、代码，几乎所有代码都给出了具体下载地址，如果在操作过程中发现下载失败或者链接失效，请联系我，可以加我的微信 peter_jw212，也可以关注微信公众号“编码美丽”进行留言，或者访问我的博客 http://blog.csdn.net/jiangwei0910410003，以及我的个人网站 http://www.wjdiankong.cn。有任何问题都可以通过这些渠道联系我，我将尽力给出详细解答。

有的读者读完这本书之后，可能发现有些内容对新技术不再生效了，比如加固和脱壳技术。我在此要说明一下，这些技术是时刻都在变的，所谓道高一尺，魔高一丈，攻防技术每一天都可能改变。所以本书选择了最基本的入门技术进行讲解，因为只有掌握了这些技术，才能继续学习并产生灵感来应对日后变化的技术。我认为最基本的入门技术也是最重要的技术。

致谢

这本书的写作历时一年多，真心觉得很不容易。如果觉得本书写得好，就请推广点赞；

如果发现本书有错误的地方，还请批评指正。毕竟第一次写书没有那么完美，期待读者的指正和批评。最后想感谢一些人，他们在我写书过程中给予了技术和精神上的支持。非常感谢非虫大神（《 Android 软件安全和逆向分析》等书作者）对本书第 4 章、第 7 章的 so 和 dex 文件格式解析技术的支持；感谢看雪论坛的 MindMac 大神对第 5 章的 arsc 文件格式解析技术的支持；感谢看雪论坛的 ThomasKing 大神对第 14 章加固技术的支持；感谢我的同学汪恒和殷传宝对我从开始写作到出版这一路上的陪伴和鼓励。

目　录

对本书的赞誉
前言

基础篇

第1章　Android中锁屏密码加密算法分析 … 2
1.1　锁屏密码方式 … 2
1.2　密码算法分析 … 2
1.2.1　输入密码算法分析 … 2
1.2.2　手势密码算法分析 … 7
1.3　本章小结 … 9

第2章　Android中NDK开发 … 10
2.1　搭建开发环境 … 10
2.1.1　Eclipse 环境搭建 … 10
2.1.2　Android Studio 环境搭建 … 12
2.2　第一行代码：HelloWorld … 14
2.3　JNIEnv 类型和 jobject 类型 … 18
2.3.1　JNIEnv 类型 … 19
2.3.2　jobject 参数 obj … 19
2.3.3　Java 类型和 native 中的类型映射关系 … 19
2.3.4　jclass 类型 … 19
2.3.5　native 中访问 Java 层代码 … 20
2.4　JNIEnv 类型中方法的使用 … 21
2.4.1　native 中获取方法的 Id … 22
2.4.2　Java 和 C++ 中的多态机制 … 24
2.5　创建 Java 对象及字符串的操作方法 … 27
2.5.1　native 中创建 Java 对象 … 27
2.5.2　native 中操作 Java 字符串 … 28
2.6　C/C++ 中操作 Java 中的数组 … 32
2.6.1　操作基本类型数组 … 32
2.6.2　操作对象类型数组 … 33
2.7　C/C++ 中的引用类型和 ID 的缓存 … 36
2.7.1　引用类型 … 36
2.7.2　缓存方法 … 37
2.8　本章小结 … 38

第3章　Android中开发与逆向常用命令总结 … 39
3.1　基础命令 … 39
3.2　非 shell 命令 … 40
3.3　shell 命令 … 45
3.4　操作 apk 命令 … 49
3.5　进程命令 … 50
3.6　本章小结 … 52

第4章　so文件格式解析 … 53
4.1　ELF 文件格式 … 53
4.2　解析工具 … 54
4.3　解析 ELF 文件 … 57
4.4　验证解析结果 … 60
4.5　本章小结 … 61

第5章　AndroidManifest.xml文件格式解析 … 62
5.1　格式分析 … 62

5.2 格式解析 …… 63
5.2.1 解析头部信息 …… 63
5.2.2 解析 String Chunk …… 63
5.2.3 解析 ResourceId Chunk …… 68
5.2.4 解析 Start Namespace Chunk …… 70
5.2.5 解析 Start Tag Chunk …… 72
5.3 本章小结 …… 82

第6章 resource.arsc文件格式解析 …… 83

6.1 Android 中资源文件 id 格式 …… 83
6.2 数据结构定义 …… 85
6.2.1 头部信息 …… 85
6.2.2 资源索引表的头部信息 …… 85
6.2.3 资源项的值字符串资源池 …… 86
6.2.4 Package 数据块 …… 87
6.2.5 类型规范数据块 …… 88
6.2.6 资源类型项数据块 …… 89
6.3 解析代码 …… 93
6.3.1 解析头部信息 …… 93
6.3.2 解析资源字符串内容 …… 94
6.3.3 解析包信息 …… 96
6.3.4 解析资源类型的字符串内容 …… 97
6.3.5 解析资源值字符串内容 …… 98
6.3.6 解析正文内容 …… 99
6.4 本章小结 …… 105

第7章 dex文件格式解析 …… 106

7.1 dex 文件格式 …… 106
7.2 构造 dex 文件 …… 107
7.3 解析数据结构 …… 108
7.3.1 头部信息 Header 结构 …… 108
7.3.2 string_ids 数据结构 …… 112
7.3.3 type_ids 数据结构 …… 115
7.3.4 proto_ids 数据结构 …… 116
7.3.5 field_ids 数据结构 …… 118
7.3.6 method_ids 数据结构 …… 119
7.3.7 class_defs 数据结构 …… 120
7.4 解析代码 …… 128
7.4.1 解析头部信息 …… 128
7.4.2 解析 string_ids 索引区 …… 129
7.4.3 解析 type_ids 索引区 …… 130
7.4.4 解析 proto_ids 索引区 …… 130
7.4.5 解析 field_ids 索引区 …… 131
7.4.6 解析 method_ids 索引区 …… 132
7.4.7 解析 class_def 区域 …… 132
7.5 本章小结 …… 134

防护篇

第8章 Android应用安全防护的基本策略 …… 136

8.1 混淆机制 …… 136
8.1.1 代码混淆 …… 136
8.1.2 资源混淆 …… 136
8.2 签名保护 …… 138
8.3 手动注册 native 方法 …… 140
8.4 反调试检测 …… 144
8.5 本章小结 …… 145

第9章 Android中常用权限分析 …… 147

9.1 辅助功能权限 …… 147
9.2 设备管理权限 …… 148
9.3 通知栏管理权限 …… 149
9.4 VPN 开发权限 …… 149
9.5 本章小结 …… 150

第10章 Android中的run-as命令 …… 151

10.1 命令分析和使用 …… 151
10.2 Linux 中的 setuid 和 setgid 概念 …… 159
10.3 Android 中 setuid 和 setgid 的使用场景 …… 162
10.4 run-as 命令的作用 …… 165
10.5 调用系统受 uid 限制的 API …… 166

10.6 本章小结 …… 168

第11章 Android中的allowBackup属性 … 169

11.1 allowBackup 属性介绍 …… 169
11.2 如何获取应用隐私数据 …… 170
11.3 如何恢复应用数据 …… 175
11.4 本章小结 …… 175

第12章 Android中的签名机制 …… 176

12.1 基本概念 …… 176
12.2 Android 中签名流程 …… 182
12.3 Android 中为何采用这种签名机制 … 191
12.4 本章小结 …… 192

第13章 Android应用加固原理 …… 193

13.1 加固原理解析 …… 193
13.2 案例分析 …… 195
13.3 运行项目 …… 206
13.4 本章小结 …… 208

第14章 Android中的so加固原理 …… 209

14.1 基于对 so 中的 section 加密实现 so 加固 …… 209
14.1.1 技术原理 …… 209
14.1.2 实现方案 …… 210
14.1.3 代码实现 …… 210
14.1.4 总结 …… 220
14.2 基于对 so 中的函数加密实现 so 加固 …… 221
14.2.1 技术原理 …… 221
14.2.2 实现方案 …… 223
14.2.3 代码实现 …… 224
14.3 本章小结 …… 230

工具篇

第15章 Android逆向分析基础 …… 232

15.1 逆向工具 …… 232
15.2 逆向基本知识 …… 233
15.3 打开系统调试总开关 …… 233
15.4 本章小结 …… 237

第16章 反编译神器apktool和Jadx …… 238

16.1 逆向操作惯例 …… 238
16.2 反编译常见的问题 …… 238
16.3 分析 apktool 的源码 …… 240
16.4 解决常见问题 …… 249
16.5 apktool 的回编译源码分析 …… 254
16.6 Jadx 源码分析 …… 256
16.7 本章小结 …… 258

第17章 Hook神器Xposed …… 259

17.1 安装教程 …… 259
17.2 环境搭建 …… 259
17.3 编写模块功能 …… 260
17.4 运行模块 …… 264
17.5 本章小结 …… 265

第18章 脱壳神器ZjDroid …… 266

18.1 ZjDroid 原理分析 …… 266
18.2 工具命令分析 …… 268
18.3 工具日志信息 …… 272
18.4 工具用法总结 …… 272
18.5 工具使用案例 …… 273
18.6 本章小结 …… 276

第19章 Native层Hook神器 Cydia Substrate …… 277

19.1 环境搭建 …… 277
19.2 Hook Java 层功能 …… 277
19.3 Hook Native 层功能 …… 279
19.4 框架使用事项说明 …… 283
19.5 本章小结 …… 283

操作篇

第20章 静态方式逆向应用 …… 286

20.1 smali 语法 …… 286
20.2 手动注入 smali 语句 …… 288
20.3 ARM 指令 …… 288
20.4 用 IDA 静态分析 so 文件 …… 290
20.5 案例分析 …… 292
20.5.1 静态分析 smali 代码 …… 292
20.5.2 静态分析 native 代码 …… 300
20.6 本章小结 …… 303

第21章 动态调试smali源码 …… 304

21.1 动态调试步骤 …… 304
21.2 案例分析 …… 305
21.3 本章小结 …… 324

第22章 IDA工具调试so源码 …… 325

22.1 IDA 中的常用快捷键 …… 325
22.2 构造 so 案例 …… 331
22.3 逆向 so 文件 …… 332
22.3.1 获取应用的 so 文件 …… 332
22.3.2 用 IDA 进行调试设置 …… 336
22.3.3 IDA 调试的流程总结 …… 343
22.4 用 IDA 解决反调试问题 …… 343
22.5 本章小结 …… 355

第23章 逆向加固应用 …… 356

23.1 逆向加固应用的思路 …… 356
23.2 获取解密之后的 dex 文件 …… 359
23.3 分析解密之后的 dex 文件内容 …… 364
23.4 逆向方法 …… 369
23.5 逆向测试 …… 373
23.6 逆向加固应用的方法总结 …… 374
23.7 本章小结 …… 376

第24章 逆向应用经典案例分析 …… 377

24.1 加壳原理分析 …… 377
24.2 脱壳过程 …… 380
24.3 如何还原应用 …… 393
24.4 脱壳经验总结 …… 394
24.5 本章小结 …… 395

第25章 Android中常见漏洞分析 …… 396

25.1 解压文件漏洞分析 …… 396
25.1.1 漏洞场景 …… 396
25.1.2 漏洞原因分析 …… 398
25.1.3 漏洞案例分析 …… 398
25.1.4 漏洞修复 …… 399
25.1.5 漏洞总结 …… 400
25.2 录屏授权漏洞分析 …… 400
25.2.1 漏洞场景 …… 400
25.2.2 漏洞原因分析 …… 402
25.2.3 漏洞修复 …… 402
25.2.4 漏洞总结 …… 403
25.3 本章小结 …… 403

第26章 文件加密病毒Wannacry样本分析 …… 404

26.1 病毒样本分析 …… 404
26.2 获取密码 …… 405
26.3 文件解密 …… 409
26.4 病毒分析报告 …… 414
26.5 本章小结 …… 414

基 础 篇

第 1 章　Android 中锁屏密码加密算法分析
第 2 章　Android 中 NDK 开发
第 3 章　Android 中开发与逆向常用命令总结
第 4 章　so 文件格式解析
第 5 章　AndroidManifest.xml 文件格式解析
第 6 章　resource.arsc 文件格式解析
第 7 章　dex 文件格式解析

第 1 章

Android 中锁屏密码加密算法分析

为了安全，Android 设备在解锁屏幕时会有密码输入，那么这个密码存放在哪里？是否为明文存储？如果是加密存储，那么加密算法是什么？这就是本章要介绍的内容。本章的目的是带领读者踏入移动安全的大门。

1.1 锁屏密码方式

Android 中现在支持的锁屏密码主要有两种：一种是手势密码，也就是常见的九宫格密码图；一种是输入密码，分为 PIN 密码和复杂字符密码，而 PIN 密码就是数字密码，比较简单。当然现在也有一个高级的指纹密码，这不是本章分析的范围，本章只分析手势密码和输入密码。

1.2 密码算法分析

在设置锁屏密码界面，用工具获取当前的 View 类，然后一步一步跟入，最终会跟到一个锁屏密码工具类：LockPatternUtils.java。每个版本可能实现逻辑不一样，这里用 5.1 版本的源码进行分析。

1.2.1 输入密码算法分析

找到源码之后，首先来分析一下输入密码算法：

```
public byte[] passwordToHash(String password, int userId) {
    if (password == null) {
        return null;
    }
    String algo = null;
    byte[] hashed = null;
    try {
            byte[] saltedPassword = (password + getSalt(userId)).getBytes();
            byte[] sha1 = MessageDigest.getInstance(algo = "SHA-1").
        digest(saltedPassword);
            byte[] md5 = MessageDigest.getInstance(algo = "MD5").
```

```
            digest(saltedPassword);
                hashed = (toHex(sha1) + toHex(md5)).getBytes();
            } catch (NoSuchAlgorithmException e) {
                Log.w(TAG, "Failed to encode string because of missing algorithm: "
        + algo);
            }
            return hashed;
        }
```

可以看到有一个方法 passwordToHash，参数为用户输入的密码和当前用户对应的 id，一般设备不会有多个用户，所以这里的 userId 是默认值 0。下面就是最为核心的加密算法了：原文密码 + 设备的 salt 值，然后分别进行 MD5 和 SHA-1 操作，转化成 hex 值再次拼接，得到的就是最终保存到本地的加密密码内容。而这里最重要的是如何获取设备对应的 salt 值，这可以一步一步跟踪代码：

```
private String getSalt(int userId) {
    long salt = getLong(LOCK_PASSWORD_SALT_KEY, 0, userId);
    if (salt == 0) {
        try {
            salt = SecureRandom.getInstance("SHA1PRNG").nextLong();
            setLong(LOCK_PASSWORD_SALT_KEY, salt, userId);
            Log.v(TAG, "Initialized lock password salt for user: " + userId);
        } catch (NoSuchAlgorithmException e) {
            throw new IllegalStateException("Couldn't get SecureRandom number", e);
        }
    }
    return Long.toHexString(salt);
}
```

查看 getSalt 方法，首先根据字段 key 为 lockscreen.password_salt 从一个地方获取 salt 值，如果发现这个值为 0，就随机生成一个，然后将其保存到那个地方，最后将 salt 转化成 hex 值即可。现在需要找到这个地方，继续跟踪代码：

```
private long getLong(String secureSettingKey, long defaultValue, int userHandle) {
        try {
            return getLockSettings().getLong(secureSettingKey, defaultValue,
    userHandle);
    } catch (RemoteException re) {
            return defaultValue;
        }
}
```

猜想应该是保存到一个数据库中了，继续跟踪代码：

```
private ILockSettings getLockSettings() {
    if (mLockSettingsService == null) {
    ILockSettings service = ILockSettings.Stub.asInterface(
                ServiceManager.getService("lock_settings"));
        mLockSettingsService = service;
    }
    return mLockSettingsService;
}
```

通过在 ServiceManager 中获取一个服务来进行操作，在 Android 中，像这种获取服务的方式最终实现逻辑都是在 XXXService 类中，这里是 LockSettingsService.java 类，找到这个类，查看它的 getLong 方法：

```
@Override
    public long getLong(String key, long defaultValue, int userId) throws
        RemoteException {
        checkReadPermission(key, userId);
        String value = mStorage.readKeyValue(key, null, userId);
        return TextUtils.isEmpty(value) ? defaultValue : Long.parseLong(value);
    }
```

其实到这里就可以看出，非常肯定是数据库中保存的，继续跟踪代码：

```
private final LockSettingsStorage mStorage;

    private LockPatternUtils mLockPatternUtils;
    private boolean mFirstCallToVold;

    public LockSettingsService(Context context) {
        mContext = context;
        // Open the database

        mLockPatternUtils = new LockPatternUtils(context);
        mFirstCallToVold = true;

        IntentFilter filter = new IntentFilter();
        filter.addAction(Intent.ACTION_USER_ADDED);
        filter.addAction(Intent.ACTION_USER_STARTING);
        mContext.registerReceiverAsUser(mBroadcastReceiver, UserHandle.ALL,
            filter, null, null);

        mStorage = new LockSettingsStorage(context, new LockSettingsStorage.
            Callback() {
            @Override
            public void initialize(SQLiteDatabase db) {
                // Get the lockscreen default from a system property, if available
                boolean lockScreenDisable = SystemProperties.getBoolean(
                        "ro.lockscreen.disable.default", false);
                if (lockScreenDisable) {
                    mStorage.writeKeyValue(db,
                        LockPatternUtils.DISABLE_LOCKSCREEN_KEY, "1", 0);
                }
            }
        });
    }
```

果然发现是保存到一个数据库中，继续查看 LockSettingsStorage.java 类：

```
class DatabaseHelper extends SQLiteOpenHelper {
    private static final String TAG = "LockSettingsDB";
    private static final String DATABASE_NAME = "locksettings.db";

    private static final int DATABASE_VERSION = 2;

    private final Callback mCallback;
```