一种基于指数函数的安卓应用安全系数计算模型

杨荣锋
西安电子科技大学
2018 年 11 月 7 日
yelbee@qq.com

摘要 该论文建立了一个客观、准确评估应用安全性的模型，主要用于分析 Android APP 扫描后的数据，根据风险的数量、分布和威胁程度，计算应用的安全分值。该模型认为，应用风险是造成应用安全问题的最主要的原因，将风险分为高危、中危、低危、警告、提醒和安全六个等级，每个等级赋予一定的分值，计算所有存在风险的分值之和，利用给定的指数函数模型，计算出该应用的安全分值，完成安全性评估。

关键词 指数函数、安全系数、安卓、应用、模型、风险等级、安全评估

一、模型描述

我们将风险分为六个等级，每个等级对应的分值见下表：

风险等级	分值	符号
高危	16	R_4
中危	8	R_3
低危	4	R_2
警告	2	R_1
提醒	1	R_0
安全	0	$R_\emptyset$

表 1.1 风险等级分值表

显然，分值分布符合 2 的指数幂的关系，即

$$R_4 = 16 = 2^4$$
$$R_3 = 8 = 2^3$$
$$R_2 = 4 = 2^2$$
$$R_1 = 2 = 2^1$$
$$R_0 = 1 = 2^0$$

特别地，令$R_\emptyset = 0$

在论文的后面，我会讲述为什么要将等级的分值进行这样的设定，此处暂且不解释。

该模型将所有的检测项目分为 12 类，其中前 3 类是应用的基本信息共 6 项，不计入风险的评估环节，从第 4 类到第 12 类总共有 68 项，每项的风险等级在评估前可能有多个取值，但是在评估后都是唯一的，例如第 4 检查项，根据应用风险的暴露程度，可能是高危，也可能是提醒，但是在评估后，可以确认为是高危和提醒的其中一个，这是唯一的。

12 类风险扫描项涵盖了大部分安卓应用存在的风险点，更详细的信息可参见论文的附录，里边有关于每一个检测项的详细说明。

第 1 类 文件信息
第 2 类 权限信息检测
第 3 类 四大组件
第 4 类 Menifest 文件检测
第 5 类 组件安全检测
第 6 类 Webview 组件安全检测
第 7 类 Sqlite 安全检测
第 8 类 网络通信安全检测
第 9 类 弱加密风险检测
第 10 类 数据安全检测
第 11 类 敏感函数调用检测
第 12 类 系统漏洞检测

检查项	类	基本信息描述
1	1	文件名、文件大小、MD5、包名、Main Activity、Min SDK、Target SDK
2	2	权限信息检测
3	3.1	Activity 组件
4	3.2	Service 组件
5	3.3	BroadcastReceiver 组件
6	3.4	ContentProvider 组件

表 1.2 基本信息表

*定义P_i为**评估前**第 i 个检测项的风险等级的**取值集合，**其中$i \in N, 1 \leq i \leq 68$，$P_1 \sim P_{68}$的取值集合均列在下表中

检查项	类	风险描述	*风险等级集合P_i
1	4.1	AndroidManifest 文件中 PermissionGroup 检测	$\{R_0, R_\emptyset\}$
2	4.2	AndroidManifest 文件中系统权限使用检测	$\{R_0, R_\emptyset\}$
3	4.3	AndroidManifest 文件中 ProtectionLevel 权限检测	$\{R_0, R_\emptyset\}$
4	4.4	AndroidManifest sharedUserId 检测	$\{R_4, R_0, R_\emptyset\}$
5	4.5	allowBackup 标志检测	$\{R_2, R_\emptyset\}$
6	4.6	Debuggable 配置检测	$\{R_4, R_\emptyset\}$
7	4.7	非必要权限检测	$\{R_0, R_\emptyset\}$
8	4.8	app 最低版本检测	$\{R_0, R_\emptyset\}$
9	5.1	Activity、activity-alias、service、receiver 组件导出检测	$\{R_3, R_\emptyset\}$

10	5.2	ContentProvider 组件导出检测	$\{R_3, R_\emptyset\}$
11	5.3	ContentProvider 目录遍历漏洞检测	$\{R_0, R_\emptyset\}$
12	5.4	Implicit Service 漏洞检测	$\{R_3, R_\emptyset\}$
13	5.5	Provider: grant-uri-permission 属性检测	$\{R_0, R_\emptyset\}$
14	5.6	Intent-Based 攻击检测	$\{R_2, R_\emptyset\}$
15	5.7	Intent Scheme URL 漏洞攻击检测	$\{R_4, R_\emptyset\}$
16	5.8	应用本地拒绝服务器漏洞检测	$\{R_2, R_\emptyset\}$
17	5.9	manifest 中定义组件未实现检测	$\{R_3, R_\emptyset\}$
18	5.10	Debug 或 Test 敏感测试组件泄露检测	$\{R_3, R_2, R_\emptyset\}$
19	5.11	Intent 不安全反射风险检测	$\{R_2, R_\emptyset\}$
20	6.1	Webview 远程执行漏洞检测	$\{R_0, R_\emptyset\}$
21	6.2	WebView 潜在 XSS 攻击检测	$\{R_0, R_\emptyset\}$
22	6.3	WebView 本地文件访问漏洞检测	$\{R_4, R_\emptyset\}$
23	6.4	WebView 密码明文存储漏洞检测	$\{R_0, R_\emptyset\}$
24	6.5	主机名弱校验检测	$\{R_3, R_\emptyset\}$
25	6.6	证书弱校验检测	$\{R_3, R_\emptyset\}$
26	6.7	中间人攻击漏洞检测	$\{R_3, R_\emptyset\}$
27	6.8	WebView 不校验证书漏洞检测	$\{R_3, R_\emptyset\}$
28	6.9	WebView 组件系统隐藏接口未移除漏洞	$\{R_2, R_\emptyset\}$
29	7.1	SQLite 数据库加密(SQLCipher)检测	$\{R_0, R_\emptyset\}$
30	7.2	SQLite 数据库加密拓展(SQLite Encryption Extension,SEE)检测	$\{R_0, R_\emptyset\}$
31	7.3	SQLite 数据库的对称密钥检测	$\{R_0, R_\emptyset\}$
32	7.4	SQLite Database Transaction Deprecated（SQL 注入）检测	$\{R_4, R_3, R_2, R_1, R_0, R_\emptyset\}$
33	7.5	Databases 任意读写漏洞检测	$\{R_3, R_\emptyset\}$
34	8.1	SSL 不安全组件检测	$\{R_0, R_\emptyset\}$
35	8.2	SSL 连接检测	$\{R_0, R_\emptyset\}$
36	8.3	HttpHost 检测	$\{R_0, R_\emptyset\}$
37	8.4	HttpURLConnection 漏洞检测	$\{R_1, R_\emptyset\}$
38	8.5	网络端口开放威胁检测	$\{R_1, R_\emptyset\}$
39	9.1	弱加密算法风险检测	$\{R_1, R_\emptyset\}$
40	9.2	不安全的密钥长度风险检测	$\{R_1, R_\emptyset\}$
41	9.3	ECB 弱加密模式风险检测	$\{R_1, R_\emptyset\}$
42	9.4	IVParameterSpec 不安全初始化向量风险检测	$\{R_1, R_\emptyset\}$
43	9.5	RSA 中不使用 Padding 风险检测	$\{R_1, R_\emptyset\}$
44	9.6	检测 keystore 是否使用密码保护	$\{R_4, R_\emptyset\}$
45	10.1	敏感信息检测	$\{R_0, R_\emptyset\}$
46	10.2	剪贴板敏感信息泄露风险检测	$\{R_0, R_\emptyset\}$
47	10.3	Intent 敏感数据泄露风险检测	$\{R_0, R_\emptyset\}$
48	10.4	PendingIntent 误用风险	$\{R_3, R_\emptyset\}$
49	10.5	密钥硬编码风险检测	$\{R_0, R_\emptyset\}$
50	10.6	数据或程序加载检查	$\{R_0, R_\emptyset\}$

51	10.7	BASE64 安全检测	$\{R_0, R_\emptyset\}$
52	10.8	文件全局读写漏洞检测	$\{R_3, R_\emptyset\}$
53	10.9	日志泄露风险检测	$\{R_0, R_\emptyset\}$
54	10.10	外部加载 Dex 检测	$\{R_4, R_\emptyset\}$
55	10.11	外部存储路径检测	$\{R_0, R_\emptyset\}$
56	11.1	安全相关的函数检测	$\{R_0, R_\emptyset\}$
57	11.2	安全相关的类检测	$\{R_0, R_\emptyset\}$
58	11.3	运行命令检测	$\{R_0, R_\emptyset\}$
59	11.4	Native Library 加载检测	$\{R_0, R_\emptyset\}$
60	11.5	外部动态加载 DEX 检测	$\{R_0, R_\emptyset\}$
61	11.6	root 代码检测	$\{R_0, R_\emptyset\}$
62	11.7	获取 IMEI 和 Device ID 敏感信息代码检测	$\{R_0, R_\emptyset\}$
63	11.8	发送 SMS 敏感代码检测	$\{R_0, R_\emptyset\}$
64	11.9	文件删除代码检测	$\{R_0, R_\emptyset\}$
65	11.10	signature 代码检测	$\{R_0, R_\emptyset\}$
66	12.1	fragment 注入漏洞检测	$\{R_3, R_\emptyset\}$
67	12.2	sqlite 数据库日志泄露漏洞检测	$\{R_2, R_\emptyset\}$
68	12.3	随机数生成漏洞检测	$\{R_4, R_\emptyset\}$

表 1.3 风险评估环节表

现在考虑评估完成后的情况，每个检测项都会有唯一标定的风险等级，即从每一项的风险等级集合P_i中选取一个合适的值，作为该检测项的评估结果。

定义$Q_i[R_x]$为**评估后**第 i 个检测项的风险等级为R_x，风险等级的分值总和为 S。其中，$x \in \{4,3,2,1,0,\emptyset\}$即表示高危、中危、低危、警告、提醒和安全六个等级，$i \in N, 1 \le i \le 68$

令

$$Q[R_x] = R_x$$

则

$$S = \sum_{i=1}^{68} Q_i[R_x]$$

显然，根据前方的定义，我们可以确认风险等级的总和 S 的取值范围为

$$0 \le S \le \sum_{i=1}^{68} MAX(P_i)$$

计算得

$$0 \le S \le 304$$

最后，定义安全系数 K

$$K(s) = \mathrm{Floor}(0.985^S \times 100)$$

其中 Floor 是向下取整函数，则 K 就是我们所要的结果

为什么采用$K(s)$来表示威胁与安全系数的关系是接下来要说明的，我们来考虑函数$K(s)$的性质：

性质一　*K(s)*是单调递减函数
性质二　*K(s)*变化率随 *S* 的增大而减小，表现为曲线由抖变缓
性质三　*K(s)*的值域为(0,100]，取值为整数，K(0) = 100, K(304) = 1
性质四　令 K(s) = 90, s = 7；令 K(s) = 70, s = 24
　　令 K(s) = 50, s = 46；令 K(s) = 30, s = 80

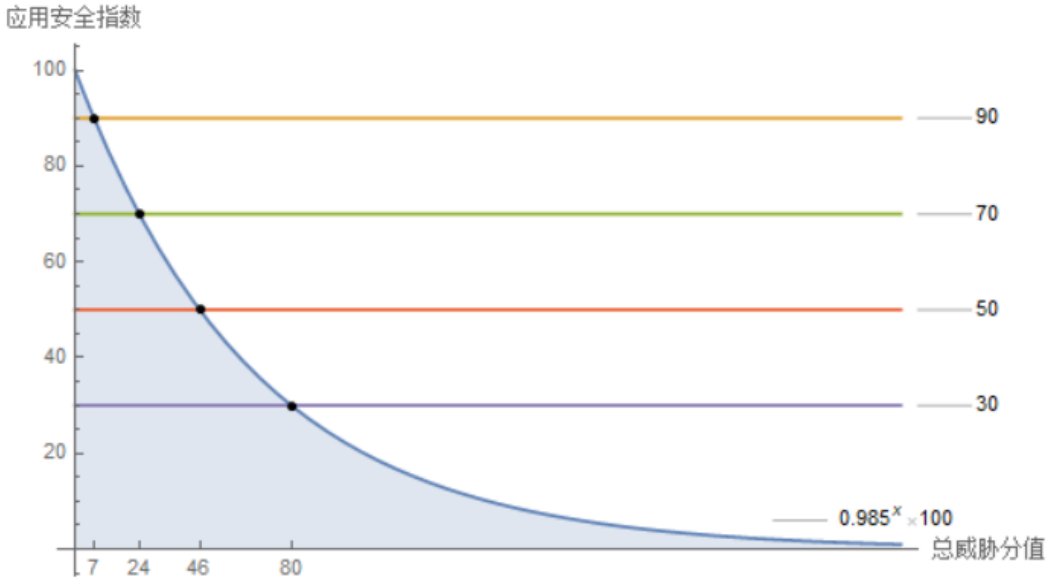


图 1.1 *K(s)*函数图像

我们可以观察到，*K(s)*符合现实中风险与安全系数的关系：

性质一

风险分值 *S* 越高，安全指数 *K* 越低，应用面临的风险越大；风险分值 *S* 越低，安全指数 *K* 越高，应用越安全。

性质二

K 前期变化率大，后期变化率小，说明在应用出现风险时，应用安全系数会快速下跌，此时往往一两个高危漏洞就可以让系统沦陷；可是当风险已经足够多时，系统已不再安全，应用已经有够多的漏洞可以利用时，再增加风险的分值，对安全系数的减小的影响会降低

性质三

风险评估采用常见的百分制。当风险分值为 0，安全系数为 100；当风险值达到最大值 304，安全系数为 1

性质四

当风险分值为 7，安全系数为 90；
当风险值为 24，安全系数为 70；
当风险值为 46，安全系数为 50；
当风险值为 80，安全系数为 30

根据性质四，我们可以制定应用安全的等级的标准

应用安全等级	安全系数分值区间
安全	90~100
合格	70~89
警告	50~69
危险	<50

表 1.4 应用安全的等级标准说明

然后，我需要说明风险等级分值标定的原因，安全可以认为对风险的贡献为 0，而随着风险等级的提升，其危害程度不应该是线性递增，而应该是指数递增，例如高危风险的危害程度应该远远大于低危风险。所以在标定风险的分值时，我采用了 2 的指数幂的方式进行标定分值，能更好地拟合现实中**危害程度**随**风险等级**的变化。

风险等级	分值	符号
高危	16	R_4
中危	8	R_3
低危	4	R_2
警告	2	R_1
提醒	1	R_0
安全	0	$R_\emptyset$

表 1.5 风险等级分值表

应用安全的等级标准分为 4 个等级，其合理与否，我们采用下表分析一下分界点 90、70、50 对应的风险组合情况

#	安全系数分值	风险分值	可能风险组合集合	风险情况
A	**90**	7	$\{R_3\}$ $\{R_2, R_1 \times 2\}$ $\{R_2 \times 2\}$ $\{R_1 \times 3\}$ $\{R_0 \times 7\}$ …	1 个中危 1 个低危 +2 个警告 2 个低危 3 个警告 7 个提醒 …
B	**70**	24	$\{R_3 \times 3\}$ $\{R_4, R_3\}$ $\{R_3 \times 2, R_2 \times 2\}$ $\{R_2 \times 6\}$ …	3 个中危 1 个高危 +1 个中危 2 个中危 +2 个低危 6 个低危 …
C	**50**	46	$\{R_4 \times 3\}$ $\{R_4 \times 2, R_3 \times 2\}$ $\{R_2 \times 12\}$ …	3 个高危 2 个高危 +2 个中危 12 个低危 …

表 1.6 风险组合与安全系数的关系

当风险组合等于或少于 A 时，应用安全等级为安全；
当风险组合位于 A 与 B 之间时，应用安全等级为合格；
当风险组合位于 B 与 C 之间时，应用安全等级为警告；
当风险组合多于 C 时，应用安全等级为危险。

二、模型评估

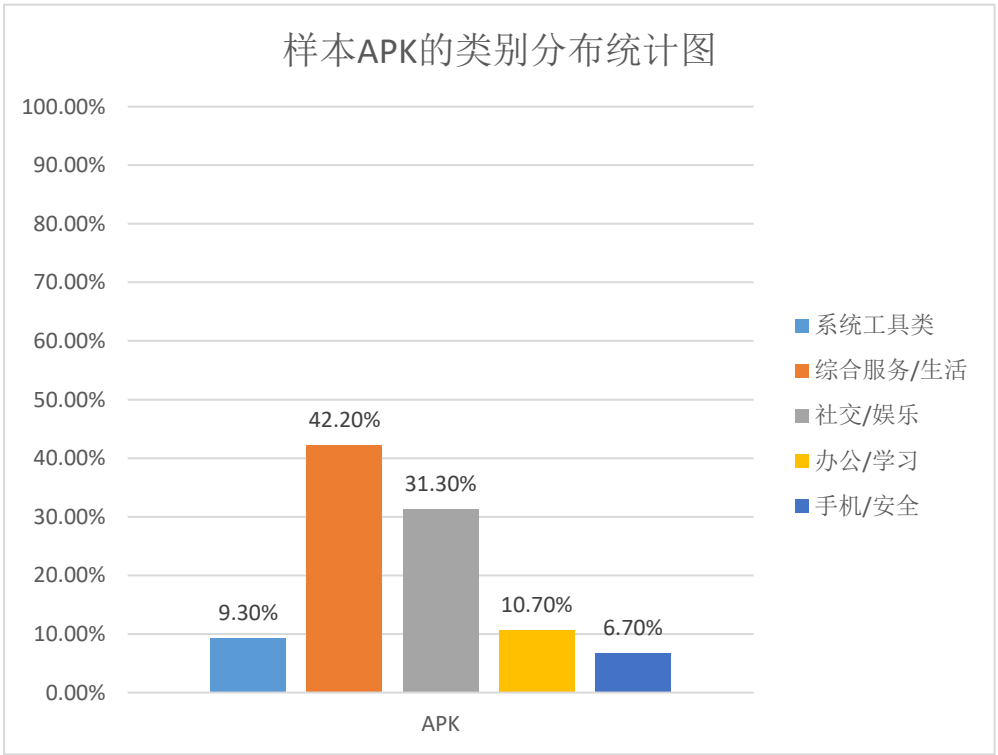
应用集合

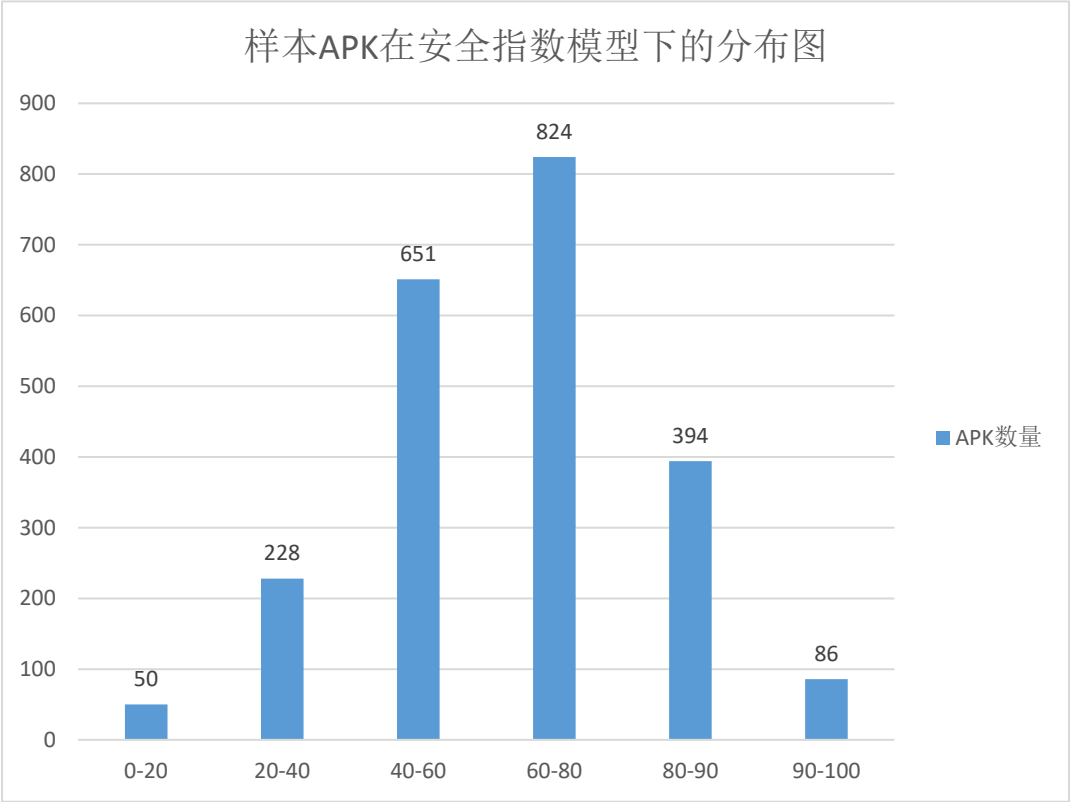
2233 个 APK 文件
总大小 29.89G
应用平均大小 13.70M

应用类别分布

我们采用抽样调查法分析应用集合中不同应用类别分布情况，从 2233 个 APK 样本中抽取了约 150 个 APK，手工安装，评估应用的类别

类别	数量	百分比
APK 总数量	3562	100.0%
能分析 APK 数量	2233/3562	62.7%
系统工具类	14/150	9.3%
综合服务/生活	63/150	42.0%
社交/娱乐	47/150	31.3%
办公/学习	16/150	10.7%
手机安全	10/150	6.7%

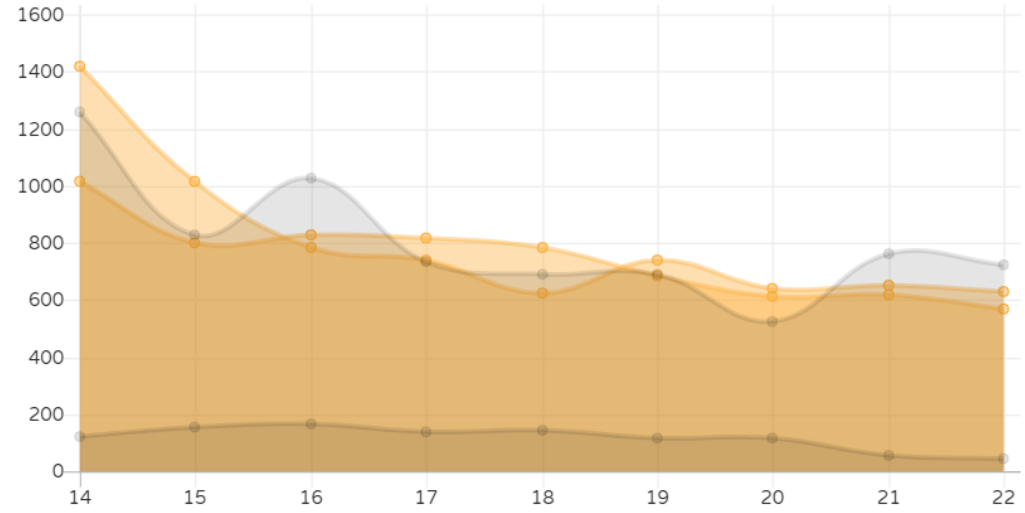




Android 版本	漏洞总数				数量
	高危	中危	低危	警告	
~~3.0-3.2(API:11-13)~~	~~1128/0.9~~	~~10247/8.2~~	~~8129/6.5~~	~~7005/5.6~~	~~1250~~
4.0-4.4(API:14-20)	957/1.3	5741/7.8	3827/5.2	5962/8.1	736
5.0-5.1(API:21-22)	99/0.4	1482/6.0	1185/4.8	1284/5.2	247

API	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**
高危	341	376	411	122	154	166	138	146	114	117	57	42
中危	3127	3415	3704	1257	825	1025	734	690	687	523	761	721
低危	2643	2709	2777	1014	799	827	815	783	682	612	619	566
警告	2335	2129	2541	1418	1017	781	738	626	740	640	653	631
检测的应用数目												

ANNHUB安全中心威胁趋势统计



ANNHUB安全中心应用威胁统计TOP-10