

基于三级签名的安卓恶意软件分析方法*

汪 欢 李学逢
(南京理工大学计算机科学与工程学院 南京 210094)

摘 要 智能手机上的恶意软件数量呈现爆炸式增长,目前迫切需要一个安全分析与检测方法。一个有效的检测与分析方法需要解决的最大问题就是如何从应用程序和恶意软件的汪洋大海中识别重新包装的应用程序。论文提出了基于三级签名的恶意软件分析方法 MSAnalytics,根据 API 调用序列,提取在操作码级的恶意软件特征,生成应用程序三级签名。实验证明,MSAnalytics 对重新包装恶意软件具有良好的分析能力。

关键词 移动智能终端;信息安全;三级签名

中图分类号 TP399 **DOI:**10. 3969/j. issn1672-9722. 2014. 08. 028

Analytic Method of Android Malware Based on Three Level Signature

WANG Huan LI Xuefeng
(Department of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing 210094)

Abstract The number of malwares on intelligent mobile phone has explosive growth, there is an urgent need to have a security analytic and detect system. An effective analytic system needs to address the biggest question: how to identify repackaged applications(or mutated malware) from the vast ocean of applications and malware. In this paper, the MSAnalytics, a three level signature based analytic method is presented to automatically analyze and extract android malware. The evaluation shows the MSAnalytics is effective in analyzing repackaging malware.

Key Words mobile intelligent terminals, information security, three level signature

Class Number TP399

1 引言

智能手机已经成为人们必备的移动设备。与此同时,智能手机上的恶意软件也以前所未有的速度出现。基于 Android 的操作系统是目前移动设备最流行的平台,同时也是恶意软件开发者瞩目的目标。如文献[1]所述,移动恶意软件呈指数性增长,主要源于恶意软件的变种。目前有很多基于权限泄漏的 Android 恶意软件检测方法,但更重要的是设计出能够综合分析恶意软件的方法,能够在操作码级别分析解剖可疑的应用程序,而不是在权限级别,并关联数据库应用程序中的恶意软件,确定它们是否属于恶意软件的突变体,并发现被感染的

合法应用程序。

为了实现上述基于 Android 移动应用的检测与分析系统,需要克服的是最大难点就是如何从应用程序和恶意软件的汪洋大海中识别重新包装的应用程序,即恶意软件变种体。据文献[2]报道,黑客可以轻松地通过注入恶意逻辑或模糊处理程序段,保留原合法应用程序的结构,同时使合法应用程序包含恶意逻辑,成为一个看似合法的恶意软件。黑客可以很容易地改变散列值或包名,所以基于加密哈希或包名的签名检测方法效果不大。因此,如何确定应用程序是否是一个重新包装或者经过伪装的恶意软件,哪个合法应用程序被感染了,是非常具有挑战性的。

* 收稿日期:2014 年 2 月 4 日,修回日期:2014 年 3 月 27 日
基金项目:国家自然科学基金项目“融合泛在网的协同防护与安全风险预测”(编号:61272419)资助。
作者简介:汪欢,女,硕士研究生,研究方向:信息安全。李学逢,男,硕士研究生,研究方向:信息安全。

2 相关工作

在安卓恶意软件数量还没有快速增长之前,研究人员更多的专注于 Android 应用程序权限和性能的泄漏。随着 Android 平台安全性危机的出现,Yajin Zhou 等^[3]首先提出了一个名为 DroidRanger 的系统,需要根据恶意软件的样本来提取每个恶意软件家族的足迹。DroidMOSS^[4]是一款 Android 系统中使用模糊散列检测重新包装的应用程序,但这个系统不适用于其他恶意软件的检测。此外,混淆技术可以改变类和方法的执行顺序,这将使得 DroidMOSS 中使用的度量产生较大的偏差。Michae 等开发了 RiskRanker^[5]来分析一个特定的应用程序是否具有危险性的行为。它使用类路径作为恶意软件家族特征,以检测更多的突变。然而,模糊处理可以很容易地沿着执行路径重新排列操作码。所以,使用类路径作为恶意软件特征不能有效地抵御混淆攻击。

对于基于 PC 的恶意软件,大量的研究工作集中在基于签名的恶意软件检测。例如,文献[6]讨论了使用签名来检测恶意软件的缺陷。文献[7]描述了隐藏恶意软件的模糊处理技术。文献[8]提出了一个自动采集恶意软件家族的恶意行为规范的系统。然而,与 PC 端相比,移动恶意软件具有不同的特点,很难将基于 PC 的恶意软件检测解决方案运用到移动设备上。

同时,重新包装是 Android 恶意软件的又一典型特点。文献[3]发现,在 Android 第三方市场有很多重新包装的应用,而且大部分都是恶意软件。

重组混淆技术是恶意软件编写者通常用来改变 apk 文件的加密哈希值,而且不会修改 dex 文件的操作码。这种技术与通过注入新的软件包到合法应用程序的重新打包技术不同。例如,使用 Java SDK 的 Jarsigner 功能重新签署 apk 文件只需要改变 apk 文件签名的一部分,保留了原始文件的逻辑和功能,并生成一个新的 apk 文件。另一个方法是使用 Apktool^[9],这是一个逆向工程工具,而且反汇编和重建 apk 文件都不需要改变任何汇编代码。即便如此,重新编译程序时还是会改变类和方法的顺序。因此,重新打包的混淆技术经常被用来根据现有恶意软件产生变异体,生成一个新版本的签名。如果反病毒系统只基于一个密码散列签名识别恶意软件,重新包装混淆技术可以很容易地逃避检测。

一个恶意软件编写者可以使用反汇编器(如

Apktool),将一个 dex 文件转换成 smali 文件,然后向 smali 代码中注入新的恶意软件逻辑,随后重建为 dex 文件。因此,恶意软件编写者可以引用各种代码混淆技术,并保留原始程序的行为,以通过反病毒检测。如文献[2]中所示,许多手机反病毒产品并不能有效地检测混淆代码体。

恶意软件编写者还可以通过互联网或附件文件动态的下载恶意代码,从而使合法应用程序具有恶意行为。另外,包内的一些附件文件也可能包含恶意逻辑。针对有效负载部分,黑客通常通过改变负载文件类型对恶意文件进行伪装。目前的检测技术只是对恶意软件是否有下载行为进行判断,并未对下载的附件进行分析。

基于上述研究,显然很需要一个更成熟的方法来检测和分析 Android 恶意软件,尤其是重新包装恶意软件。相对于直接进行恶意软件检测,本文更专注于设计一个分析方法。针对黑客可能会使用的模糊技术的鲁棒性,本文提出了一个全新的三级签名机制,以识别恶意代码。

3 MSAnalytics 的设计与实现

MSAnalytics 的体系结构如图 1 所示。

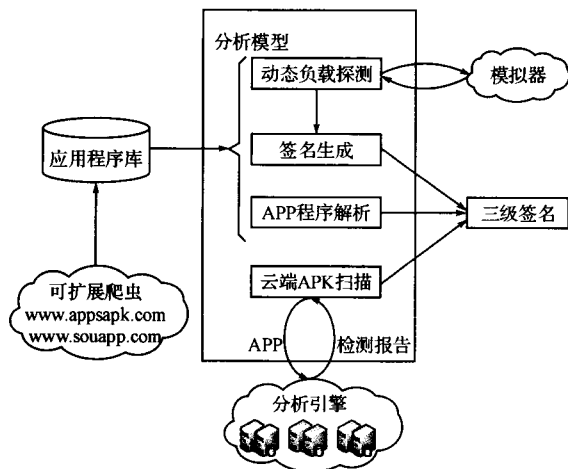


图 1 MSAnalytics 体系结构图

如图 1 所示,MSAnalytics 主要包括恶意软件自动收集模块、分析模块。其中,分析模块有包括动态负载探测、APP 程序解析、签名生成,用于生成应用程序三级签名。云端 APK 扫描器支持多种反病毒软件,其扫描结果有利于分析人员进行参考分析。

3.1 自动收集

在 MSAnalytics 中,采用了基于 Scrapy 爬虫技术^[10]来实现应用程序自动收集。用户可以指定官方或第三方安卓产品市场,爬虫工具将采用常规

的移动终端应用程序下载方式下载应用程序,从而能够系统地建立移动终端应用程序的数据库,便于系统后期进行恶意软件的分析 and 关联。截至当前,已经收集了 1,368 的移动应用程序,并进行了详细的安全分析。

3.2 动态负载探测

针对恶意软件通过互联网或附件文件动态的下载恶意代码,MSAnalytics 实现了动态载荷检测器模块,能在恶意软件包内确定恶意触发代码,并在模拟器中跟踪下载的应用程序及其行为。首先,扫描 APK 包寻找可疑文件,如. elf 或. jar 文件。黑客通常通过改变文件类型对恶意文件进行伪装。为此,该模块扫描包内所有文件,并使用魔数技术识别文件类型,而不是通过文件扩展名。其次,如果一个应用程序有任何上网行为,例如网络权限或重新委托其他应用程序下载文件^[11],动态负载探测器将把这些文件作为目标,并在模拟器中运行该应用,使用符号技术来触发下载行为。无论是包内的可疑文件还是从网络上动态下载的文件,都会被发送到签名生成器作进一步的分析。

3.3 APP 程序解析

MSAnalytics 采用 AIS(Android App Informations)数据结构,用于表示. apk 文件的信息结构。使用 APP 程序解析,可解密应用程序内的 AndroidManifest.xml 文件,分析出包名、权限信息、广播接收信息、反汇编代码等。同时通过反汇编. dex 文件生成. smali 代码,随后从. smali 源代码包中提取函数、类等信息并保存在 AIS 中,便于检索。

3.4 签名生成

反病毒公司通常使用加密散列,如 MD5,生成一个应用程序的签名。这里主要有两个缺点:首先,黑客可以很容易地变异一个应用程序,并更改其加密散列;其次,加密散列在安全性分析方面不够灵活。在 MSAnalytics 中,拟采用三层签名生成方案,以识别应用程序。该签名方案基于移动应用程序、类、方法以及恶意软件的动态负载(如果有的话)。对于任何功能的应用程序,它都需要调用各种 Android 的 API,而且方法中的 API 调用序列是很难修改,除非彻底改变了程序的逻辑。而且从目前收集到的 1368 应用程序中,没有发现任何使用这种模糊处理技术的程序。因此,MSAnalytics 中采用 API 调用序列来生成应用程序签名,通过混合不同的方法生成类的签名,应用程序的签名则由它的类的所有签名组成。需要强调的是,该签名方

法不仅可以有效抵制恶意软件混淆,更重要的是,通过基于签名的类/方法关联,有利于对恶意软件进行分析。MSAnalytics 中签名生成具体流程如下:

1) Android API 调用表:MSAnalytics 使用 Android SDK 的 API 调用表。android.jar 文件是由 Android SDK 中提供的框架包,采用 Java 反射机制^[12]取得 API 调用的所有描述。对于每个 API,提取其类路径和方法名,并给每个完整路径方法分配一个十六进制数作为 ID 的一部分。MSAnalytics 提取 Android SDK 4.1 版本中的 47,126 个完整路径方法作为其 API 调用表。表 1 为 API 调用快照表,例如,android/content/Intent;→(init)被分配的 ID 为 0x30291。

表 1 Android API 调用快照表

完整路径方法	方法 ID
android/accounts/Account;→<init>	0x00001
⋮	⋮
android/content/Intent;→<init>	0x30291
android/content/Intent;→toUri	0x30292
android/telephony/SmsManager;→getDefault	0x39D53
android/app/PendingIntent;→getBroadcast	0xF3E91

2) 反汇编:每个 Android 应用程序是由不同的类组成,而每个类又是由不同的方法构成。为给每个类或方法生成签名,MSAnalytics 首先反编译. apk 文件,找出. dex 文件的 Dalvik 操作码,并将其转变为方法和类,最后再使用 Android API 调用表来生成签名。

3) 生成 Lev3 签名(方法签名):为每个方法生成的签名即为 Lev3,也称为方法签名。基于 Android 的 API 调用表,系统提取 API 调用 ID 序列作为每个方法的字符串值,对该字符串值进行哈希算法得到方法的签名。图 2 描述了如何给将消息发送到另一移动电话的方法生成 Lev3 签名。如图 2 所示,该方法包含三个 API 调用,使用 Android API 调用表(如表 1)判断其 ID。一个方法的签名是由所有这些 API 调用的方法 ID 组成。注意,对于在运行时不会被执行的 API 调用,MSAnalytics 不会对其进行提取,因为这些代码通常都是通过模糊处理产生。此外,如果一个方法(除了 main 方法外)不会被其他任何方法调用,这个方法就可能是恶意软件编写者编写的无效方法,签名生成器也会忽略此方法。

4) 生成 Lev2 签名(包括类签名和动态负载签名):MSAnalytics 基于类里方法的 Lev3 签名,将

会为每个类生成 Lev2 签名。恶意软件编写者可以使用各种混淆或重新包装技术来改变 dex 文件中方法的调用顺序。因此,签名生成算法先将该类中的 Lev3 签名进行排序,然后串连所有 Lev3 签名,形成 Lev2 签名。

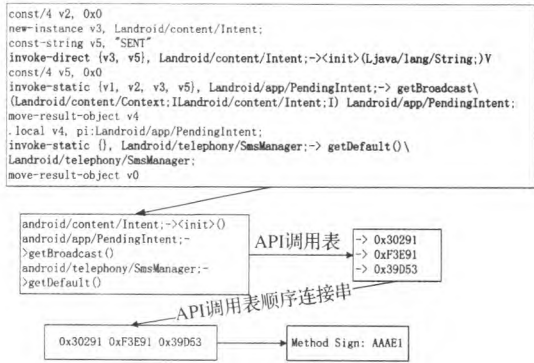


图 2 Lev3 签名生成过程图

一些恶意代码在程序执行期间动态地从互联网上进行下载行为。MSAnalytics 使用动态载荷探测组件,以获得有效载荷文件。对于动态载荷是 dex 文件或 jar 文件的,MSAnalytics 会将它们视为恶意软件中的类,并检查它们的 API 调用序列,同时为应用程序中的每个类生成一个 Lev2 签名。对于包含如 elf 文件或 so 文件的动态载荷,MSAnalytics 将它们视为恶意软件中的一个类,并使用负载的加密哈希值作为其 Lev2 签名。对于动态载荷是 apk 文件的,MSAnalytics 视每一个新的应用程序为恶意软件内的一个类,并使用该新 apk 文件的加密哈希值作为恶意软件的一个 Lev2 签名。

5) 生成 Lev1 签名(应用程序签名):该 Lev1 签名是基于 Lev2 签名的,例如,一个应用程序中的所有合格类的签名。此外,签名生成器将忽略那些不会被任何其他类调用的类(除了主类),因为这些无效的类可以通过模糊处理来产生。恶意软件编写者可以使用一些重新包装或混淆技术来改变 dex 文件类表的顺序,签名算法将先排序所有的 Lev2 签名,然后串连这些 Lev2 签名生成 Lev1 签名。

签名生成整体流程如图 3 所示。图 3 中,AAAE1 和 B23E8 的 Lev3 签名是同一个类中两个

方法签名。根据这两个已排序的 Lev3 签名,生成相应的类的 Lev2 签名 53EB3。需要注意的是 Lev2 签名 C3EB3 是从一个 dex 文件生成而来,该文件是用于执行恶意行为的动态有效载荷。根据所有类已排序的 Lev2 签名,生成应用程序的 Lev1 签名 F32DE。

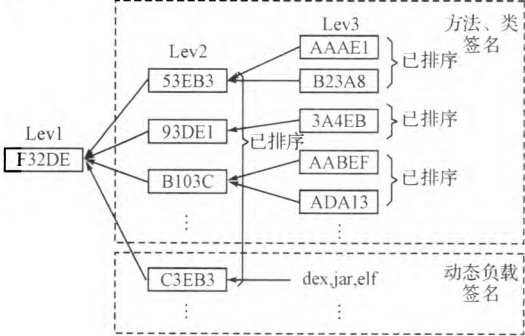


图 3 三级签名生成图

4 实验与分析

本节主要是对前文提出的基于三级签名的 Android 恶意软件分析方法 MSAnalytics 的主要功能进行实验说明,分别对重组恶意软件、代码混淆恶意软件、动态负载恶意软件进行试验。

4.1 重组恶意软件分析

MSAnalytics 可以检测由重组混淆产生的恶意软件。如果索引文件内的操作码没有修改,MSAnalytics 在生成 Lev1 签名之前先排序 Lev2 和 Lev3 签名,MSAnalytics 会产生与重组前应用程序相同的签名。

实验证明:实验结果如图 4 所示。Opfake 是一个服务器端多态性恶意软件,下载时会自动发生变异。分析对比两次下载 Opfake 时的循环冗余码(CRC)发现,唯一有意义的变化发生在位于“res/raw/”文件夹中的 data.db 文件,修改 data.db 可更改“META-INF”文件夹中包的签名数据。通过分析这种恶意软件的行为发现,在 Opfake 家族所有的突变发生在相同的操作码(存储在 classes.dex)。对此,MSAnalytics 签名系统会为这个恶意软件家族所有的变异体生成相同的 Lev1 签名。

Lev1 Signature Details of Opfake			
MDS	Package Name	Lev1 Signature	Detection Info
e70761dd3d89d26eed3932d54089a121	com.fywork.xiaohun	ec6618fc8b7de05d4b036df882e9a317	Trojan-SMS.AndroidOS.Opfake.bo
db9ae068d042e92c01fa5a26e3d434c8	com.fywork.xiaohun	ec6618fc8b7de05d4b036df882e9a317	Trojan-SMS.AndroidOS.Opfake.bo
da697083b250d80761130fe966a5d	com.fywork.xiaohun	ec6618fc8b7de05d4b036df882e9a317	Trojan-SMS.AndroidOS.Opfake.bo
e42b7361d15a29edce370372c2852e41	com.fywork.xiaohun	ec6618fc8b7de05d4b036df882e9a317	Trojan-SMS.AndroidOS.Opfake.bo
d6926a52ad43d05e0982991a5634bf32	com.fywork.xiaohun	ec6618fc8b7de05d4b036df882e9a317	Trojan-SMS.AndroidOS.Opfake.bo

图 4 Opfake 家族的 Lev1 签名的截图

图 5 所示为使用 MSAnalytics 分析 Kmin 家族的实验截图。首先计算数据库中的所有 1368 个应用的 Lev1 签名。通过计算可知,最常见的签名来自 117 个应用程序,即 Lev1 签名 90b3d4af183e9f72f818c629b486fdec,而且所有这些应用程序都具有不同的 MD5 值。这表明,传统的加密散列(如

MD5)无法识别的恶意软件变异体,但 MSAnalytics 可以有效地识别它们。此外,这 117 个应用程序都是 Kmin 家族的变种。经过进一步的分析发现,Kmin 家族是一个壁纸更新应用程序,它的所有变体具有相同的应用程序的结构和相同的恶意行为,唯一的区别是,他们有不同的图标和壁纸文件。

Lev1 Signature Details of Kmin			
MD5	Package Name	Lev1 Signature	Detection Info
ad6ef8ac95566025d5521d59ad5ae5d6	com.km.launcher	90b3d4af183e9f72f818c629b486fdec	Backdoor.AndroidOS.Kmin.d
f8a1f5cc627e0b0c54c19ab37e807a	com.km.launcher	90b3d4af183e9f72f818c629b486fdec	Backdoor.AndroidOS.Kmin.d
1f673b6a301ccc30a854943502797154	com.km.launcher	90b3d4af183e9f72f818c629b486fdec	Backdoor.AndroidOS.Kmin.d
5bd29b14b27fcd8d36ae803b8418bc7	com.km.launcher	90b3d4af183e9f72f818c629b486fdec	Backdoor.AndroidOS.Kmin.d
acc4787d443005f9c5949aa53c8d0234	com.km.launcher	90b3d4af183e9f72f818c629b486fdec	Backdoor.AndroidOS.Kmin.d
f17e520ac79bd9ec5ce7baef83011a2b	com.km.launcher	90b3d4af183e9f72f818c629b486fdec	Backdoor.AndroidOS.Kmin.d
18b80f0e9da76c3a87e7ad91fa2cae6	com.km.launcher	90b3d4af183e9f72f818c629b486fdec	Backdoor.AndroidOS.Kmin.d
5a617380d2b98b5a1f1d7f54486ebb1b	com.km.launcher	90b3d4af183e9f72f818c629b486fdec	Backdoor.AndroidOS.Kmin.d
a32d2ed06323ca791711278797a7d608	com.km.launcher	90b3d4af183e9f72f818c629b486fdec	Backdoor.AndroidOS.Kmin.d
f12238101902750604ca900932cca8cc	com.km.launcher	90b3d4af183e9f72f818c629b486fdec	Backdoor.AndroidOS.Kmin.d
1750697625b0f59c8946ba351d3a1465	com.km.launcher	90b3d4af183e9f72f818c629b486fdec	Backdoor.AndroidOS.Kmin.d
594d8403c1c3ff7b56352efb9f2e809	com.km.launcher	90b3d4af183e9f72f818c629b486fdec	Backdoor.AndroidOS.Kmin.d

图 5 Kmin 家族的 Lev1 签名的截图

4.2 代码混淆恶意软件分析

MSAnalytics 忽略在运行时不被执行的类和方法的 API 调用,因为其无效性,而且可通过模糊处理生成。同时,该签名生成不依赖于方法名或类名等,故名称混淆对该签名机制没有影响。此外,MSAnalytics 的签名生成是根据分析者自定义的 API 调用表,因此可以灵活地更新表,从而对抗不同的代码混淆。

实验证明:为了说明 MSAnalytics 对代码混淆的有效性,选择来自三个不同家族的 30 个不同的恶意软件作为样本(每个家族 10 个样品)。采用的恶意软件家族有:Basebrid(或 Basebridge)、Gold-

Dream 和 KungFu。ADAM^[13]可以通过各种重新包装和模糊处理技术(如注入新方法、修改方法名等),自动变换原始的恶意软件样本为不同的变体。本实验通过 ADAM 为每个恶意软件产生七种不同的变种。并将这 240 个新恶意软件样本放入 MSAnalytics 系统,并检查它们的签名。如表 2 结果显示,经过签名计算后,对于每一个恶意软件,原始样品及其七个突变的变种有不同的 MD5 哈希值(三个重新包装,四个代码混淆),但它们都具有相同的 Lev1 的签名。这表明,MSAnalytics 签名系统能有效地抵御代码混淆。

表 2 代码混淆示例表

包名	家族	MD5	Lev1 签名	说明
com.tutusw.phonespeedup	Kungfu	f7967f71b2f32287660ae9a3fa366022	3c83ed0f80646c8ba112cb8535c293dd	原始软件
com.tutusw.phonespeedup	Kungfu	f2e2d727f95fa868fd7ff54459e766e3	3c83ed0f80646c8ba112cb8535c293dd	重新包装
com.tutusw.phonespeedup	Kungfu	e01f573cca83fdf2737be6ecee35fe33	3c83ed0f80646c8ba112cb8535c293dd	重新包装
com.tutusw.phonespeedup	Kungfu	d383ceeb9c6ffeff8c0dd12b73ec43e3	3c83ed0f80646c8ba112cb8535c293dd	重新包装
com.tutusw.phonespeedup	Kungfu	cd040541693693faca9fec646e12e7e6	3c83ed0f80646c8ba112cb8535c293dd	代码混淆
com.tutusw.phonespeedup	Kungfu	401952d745cd7ca5281a7f08d3e2eede	3c83ed0f80646c8ba112cb8535c293dd	代码混淆
com.tutusw.phonespeedup	Kungfu	271c3965c7822ebf944feb8bbd1cfe7f	3c83ed0f80646c8ba112cb8535c293dd	代码混淆
com.tutusw.phonespeedup	Kungfu	8b12ccdc8a69cf2d6a7e6c00f698aaaa	3c83ed0f80646c8ba112cb8535c293dd	代码混淆

4.3 动态负载恶意软件分析

有些恶意软件将动态地下载包含恶意代码的文件。另外,包内的一些附件文件也可能包含恶意逻辑,但同时它们又可以被隐藏成其他有效文件

(如.png 文件和.wma 文件)。MSAnalytics 将把这些文件视为动态载荷。通过使用前文介绍的动态负载探测和签名生成,MSAnalytics 将为这些有效载荷产生不同的签名。

表 3 动态负载示例表

恶意软件	动态负载	MD5	说明
GinMaster	/assets/runme.png	34cb03276e426f8d 61e782b8435d3147	获取 root 权限
GinMaster	/assets/gb1m.png	a24d2ae57c3ceelcf 3298c856a917100	获取 root 权限
	/assets/install.png		
	/assets/installsoft.png		
	/assets/runme.png		
GinMaster	/assets/gbfa.png	9e847c9a27dc9898 825f466ea00dac81	获取 root 权限
	/assets/install.png		
Plankton	plankton v0.0.4.jar	dcbe11e5f3b82ce89 1b793ea40e4975e	运行时下载

实验证明:在恶意软件数据库中,使用 MSAnalytics 系统检测到的一些恶意软件包含相同的带有 PNG 文件扩展名的文件。但是,当检查这个文件的魔法数字时,发现它实际上是一个 .elf 文件。经过进一步的分析发现,这个文件会获取 root 权限,这个恶意软件属于 GinMaster(或 GingerMaster)家族。同时,通过使用动态分析,MSAnalytics 发现 Plankton 家族的所有恶意软件在主应用程序开始运行时,将下载 plankton_v0.0.4.jar(或类似名称的.jar)文件。进一步的分析显示,这个下载的.jar 文件包含恶意行为,例如,偷取浏览器的历史记录信息,产生桌面快捷方式和僵尸网络逻辑。表 3 是通过 MSAnalytics 系统检测到的一些有代表性的使用动态有荷的恶意软件。

5 结语

随着安卓移动智能终端的急速发展,基于安卓平台的恶意软件应运而生,发展速度惊人。本文在对移动智能终端恶意软件现有相关工作进行分析的基础上,提出了基于三级签名的安卓恶意软件分析方法 MSAnalytics,据 API 调用序列,提取在操作码级的恶意软件特征,生成应用程序三级签名。实验通过利用 MSAnalytics 对重组恶意软件、代码混淆恶意软件、动态负载恶意软件的分析,实验结果证明,MSAnalytics 具有良好的分析能力。

参考文献

[1] Dirro T, Greve P, Kashyap R, et al. McAfee threats report: second quarter 2011[J]. McAfee, Inc, 2011, 2821.

[2] Zheng M, Lee P P C, Lui J C S. Adam: An automatic and extensible platform to stress test android anti-virus systems [C]//Detection of Intrusions and Malware, and Vulnerability Assessment. Heidelberg: Springer, 2013: 82-101.

[3] Zhou Y, Wang Z, Zhou W, et al. Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets[C]//Proceedings of the 19th Annual Network and Distributed System Security Symposium, 2012.

[4] Zhou W, Zhou Y, Jiang X, et al. Detecting repackaged smartphone applications in third-party android marketplaces[C]//Proceedings of the second ACM conference on Data and Application Security and Privacy. ACM, 2012: 317-326.

[5] Grace M, Zhou Y, Zhang Q, et al. Riskranker: scalable and accurate zero-day android malware detection [C]//Proceedings of the 10th international conference on Mobile systems, applications, and services. ACM, 2012: 281-294.

[6] Moser A, Kruegel C, Kirda E. Limits of static analysis for malware detection[C]//Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual. IEEE, 2007: 421-430.

[7] O’Kane P, Sezer S, McLaughlin K. Obfuscation: The hidden malware[J]. Security & Privacy, IEEE, 2011, 9 (5): 41-47.

[8] Christodorescu M, Jha S, Kruegel C. Mining specifications of malicious behavior[C]//Proceedings of the 1st India software engineering conference. ACM, 2008: 5-14.

[9] Apktool. [OL]. <http://code.google.com/p/android-apktool/>, 2009.


[10] Scrapy. [OL]. Available: <http://www.scrapy.org>.

[11] Felt A P, Wang H J, Moshchuk A, et al. Permission Re-Delegation: Attacks and Defenses[C]//USENIX Security Symposium, 2011.

[12] Eugster P. Using JAVA reflection. [OL]. <http://docs.oracle.com/javase/tutorial/reflect/index.html>. 2011.

[13] M. Zheng, C. Lee. ADAM. [OL]. <http://ansrlab.cse.cuhk.edu.hk/software/adam/>. 2012.

基于三级签名的安卓恶意软件分析方法

作者：[汪欢](#)，[李学逢](#)，[WANG Huan](#)，[LI Xuefeng](#)
作者单位：[南京理工大学计算机科学与工程学院](#) [南京210094](#)
刊名：[计算机与数字工程](#)
英文刊名：[Computer and Digital Engineering](#)
年，卷(期)：2014, 42 (8)

引用本文格式：[汪欢](#). [李学逢](#). [WANG Huan](#). [LI Xuefeng](#) [基于三级签名的安卓恶意软件分析方法](#) [期刊论文]-[计算机与数字工程](#)
2014 (8)