

基于安全域的 Android 系统内核安全增强机制研究

陈 伟 杨秋辉 程雪梅

(四川大学计算机学院 成都 610065)

摘 要 近年来,随着 Android 系统的迅猛发展,其安全面临着极大的挑战。Android 的安全包括系统安全和软件安全,系统安全是整个安全的基石,对整体安全至关重要。以 TOMOYO Linux 为基础,针对 Android 的特点,改进了域生成算法,实现了内核层面的安全增强机制。实验证明,所提安全增强方案能够有效增强 Android 系统的安全性。

关键词 Android,安全增强,安全域,TOMOYO Linux,内核级安全

中图分类号 TP309.1 文献标识码 A DOI 10.11896/j.issn.1002-137X.2018.02.035

Study on Security Enhancement Mechanism of Android System Kernel Based on Security Domain

CHEN Wei YANG Qiu-hui CHENG Xue-mei

(College of Computer Science, Sichuan University, Chengdu 610065, China)

Abstract Android system is facing challenges on security with its explosive development recently. Android system's security is composed of system security and software security. System security is the cornerstone of integral security, which is vital to Android. In order to strengthen the system security, this paper proposed an improved domain generation algorithm to enhance the kernel level security mechanism based on TOMOYO Linux. In addition, the experiment shows that the proposed method is efficient to enhance the security of Android system.

Keywords Android, Security enhancement, Security domain, TOMOYO Linux, Kernel-level security

1 引言

近年来,在 Android 系统迅猛发展的同时,其安全性也一直备受挑战^[1]。自 2004 年第一个手机病毒被公开报道以来,恶意代码呈指数增长,它们对 Android 系统的安全性造成了极大的威胁。因此,各大安全软件企业纷纷投身 Android 安全研究,并相继推出防护软件来保护 Android 设备的安全性。然而,这些软件的防护多数停留在应用层,很难深入到内核级别提供更深层的安全保护。

目前,在系统内核安全机制中,Android 将 SELinux 引入其内核(称其为 SEAndroid),从内核级别大大加强了操作系统的安全性。但在 Android 4.2 及 Android 4.3 版本中,SEAndroid 以请求模式运行,只做记录,并不进行系统保护,在 Android 4.4 及其之后的版本中,SEAndroid 才以强制模式运行,从而真正保护内核级安全。因此,Android 4.2 之前的原生 Android 系统根本无法得到内核级别的保护;并且,SEAndroid 安全策略的制定需要操作者对 SELinux 的运行方式有较深入的理解,普通用户难以胜任。

SELinux 和 TOMOYO Linux 都可以为系统提供强制访问控制,提高系统的安全性。但在域的控制和策略的制定上,

TOMOYO Linux 比 SELinux 更容易^[2]。然而,原有的 TOMOYO Linux 域的架构粒度不够细,难以保证系统的安全性。因此,本文在原有 TOMOYO Linux 的基础上,根据 Android 的特点,从根本上改变了域架构,增强了系统对进程的控制能力,实现了系统内核层面的安全。

2 相关工作

Android 系统的安全性引起了诸多研究者的关注。在系统安全方面,一是挖掘系统漏洞非常必要。美国软件分析公司 Coverity 于 2010 年宣布了 Android 内核中大量的系统漏洞,其中有 88 个可能导致用户敏感信息泄漏的高风险漏洞;刘剑等人^[3]则开发了 Android 系统漏洞分析工具,该工具基于控制流,通过构建分析脚本,分析了 Android 内核代码的典型错误,从而检测发现了代码库中的一系列脆弱点。二是增强系统内核的安全性,避免由于漏洞造成危害同样值得关注^[4]。Shabtai 等人^[5]于 2010 年提出了通过将 SELinux 引入 Android 内核,来实现安全机制增强的思路;Bugiel 等人^[6]以 TOMOYO Linux 为基础,于 2011 年开发了 TrustDroid 系统,以增强 Android 的内核安全;Smalley 等人^[7]于 2013 年实现了基于 SELinux 内核的 Android 系统;同年,Bugiel 等人^[8]在

到稿日期:2016-11-21 返修日期:2017-02-10

陈 伟(1991-),男,硕士生,主要研究方向为嵌入式测试与开发,E-mail:carychence@qq.com;杨秋辉(1970-),女,博士,硕士生导师,主要研究方向为软件测试,E-mail:yangqiuhi@scu.edu.cn(通信作者);程雪梅(1991-),女,硕士生,主要研究方向为嵌入式测试与开发,E-mail:852627915@qq.com。

以 SELinux 为基础的 Android 4.0 系统上设计了一套协议语言,提供了定制安全策略的服务,从而使 SELinux 的安全特性在 Android 系统的架构层得以应用。Android 4.2 版本已经将 SELinux 加入到官方发布的系统中,表明更加安全的 Android 系统来临,而对 Android 系统安全的研究也将因此拥有更加广阔的发展空间。

此外,Android 安全增强机制的其他方面也引起了国内外研究者的关注。丁涓等人^[9]于 2015 年提出了一种基于远程服务控制 Binder 通信机制的安全增强机制,应用程序只能根据管理端的安全策略规定运行,对于没有的权限,将进行拦截并上报。路晔绵等人^[10]在攻击程序中 Hook 相关 API 调用,增强了在第三方推送服务的安全性。Wan 等人^[11]为了从内部和外部保护 Android 应用信息,建立了一种基于分析层级过程的评估模型。Xu 等人^[12]分析了基于 ADB 的攻击的保护作用,改进了 USB 调试模式的安全机制模式,增加了 USB 调试的安全性。

通常将引入了 SELinux 的 Android 称为 SEAndroid。SEAndroid 包含 3 种运行模式:请求模式、强制模式及关闭模式。在 Android 4.2 及 Android 4.3 版本中,SEAndroid 以请求模式运行,在 Android 4.4 及之后的版本中,SEAndroid 才以强制模式运行。到目前为止,即使在最新的 Android 5.0 版本上,强制访问控制(MAC)机制的所有功能也并未完全集成到 Android 系统中^[13]。在企业级 Android 系统安全实践中,OEM 厂商开发了相应的系统安全加固方案。比如,Samsung Knox 就是一个完整的 Android 系统安全解决方案,该方案包括安全启动、内核完整性检测及 SEAndroid 等^[14],但它只能应用于三星公司的手机上^[15]。

本文将 Android 操作系统内核级安全增强作为研究对象,对现有的安全机制及增强方案进行了研究,进而提出了一套基于 TOMOYO Linux 域架构的简单且有效的 Android 系统安全增强方案。

3 Android 系统内核安全增强机制

Android 系统的安全增强引擎共分为三大模块:域管理、安全策略库设计和决策管理。安全策略库由具体规则组成,具体规则描述了主体(安全域)对客体(各种系统资源)的权限信息;决策管理依据安全策略库做出具体裁决,改变主体对客体的行为结果;而域管理中的域即是指安全策略库规则的作用域,以表明该规则适用于何种进程。本文主要对域管理模块进行了改进,基于 TOMOYO Linux 域架构,提出了一套适合 Android 系统的域管理机制,并针对改进后的域提出了安全策略规则制定的方法,有效地增强了系统的安全性。

3.1 TOMOYO Linux 域架构

如图 1 所示,TOMOYO Linux 的域架构呈树状。按照执行过程,系统中的进程被划分到不同的域。进程的每个执行操作都会引起一次域转换。如果直接将这种架构应用到 Android 系统中,会引起两个问题。

1)在 Android 系统中,所有的应用程序都由 Zygote 生

成,都源于同一个进程 app_process。虽然这些应用程序的进程可以用不同的 uid(应用程序 ID)进行区分,但是它们的域都是<kernel> /system/bin/app_process。如果直接对该域制定规则,则该规则适用于所有源于 app_process 的进程。这一范围既包括联系人、短信和通话等系统中的重要进程,也可能含有恶意代码的第三方应用。

2)程序在运行过程中的执行操作是否发生及何时发生是不可预见的。按照 TOMOYO Linux 的域架构,进程的每个执行动作都会引起进程的域转换,使得原本处于域规则控制下的进程,随时可能因为域转换的发生而失去控制。

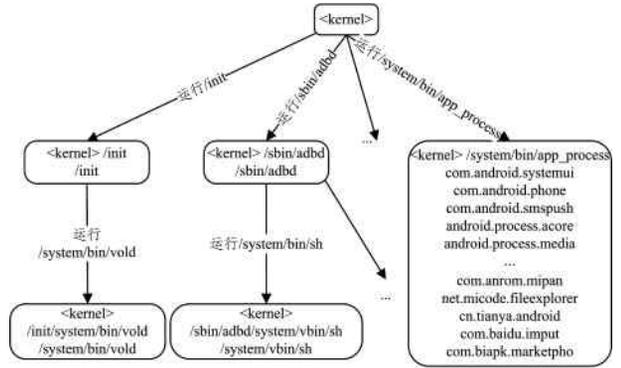


图 1 TOMOYO Linux 域架构^[16]

Fig. 1 Domain architecture of TOMOYO Linux^[16]

3.2 改进的域架构

本文针对 TOMOYO Linux 域架构的问题而提出的安全增强机制是基于一种全新的域架构的(见图 2),按照不同的应用程序将本来属于 app_process 的所有进程拆分成不同的域。与此同时,还须确保这些进程的域不会随着执行操作的进行而发生转换。这样,管理者就可以针对每个应用程序的功能和安全性分别制定访问规则。每个应用程序都拥有自己的访问规则,从而增加了安全系统的可控性。这种域架构改变了传统 TOMOYO Linux 以执行过程为唯一依据的简单域划分方法,创造了执行过程与应用程序名相结合,两者共同决定应用程序域的全新域架构。

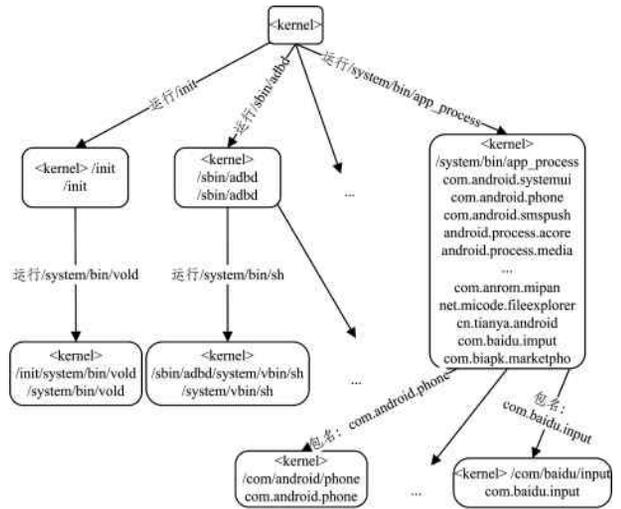


图 2 基于 TOMOYO 的 Android 域架构

Fig. 2 Android domain architecture based on TOMOYO

如图 2 所示,首先,改进的域架构需要根据域名生成规则生成域,然后要防止域之间进行转换,最后将进程固定在单独的域中。这样,系统中的每个应用程序都属于一个独立的域,该域在系统中是唯一的,不存在一个进程属于多个域或者多个进程属于一个域的情况,即进程和域的关系是一一对一的。同时,进程的域一旦生成,就不再改变,这防止了域转换造成程序失去控制情况的发生。这样的域架构设计,在很大程度上细化了安全机制的控制粒度,对提升 Android 系统的安全性有显著的效果。

3.2.1 域名生成规则

应用程序的包名由 AndroidManifest.xml 文件中的 package 字段决定,每个应用程序的该字段值都是唯一的。此外,应用程序的 ID(简称 uid)在系统中也是唯一的,但不稳定,当进程重启或手机重启后,很可能发生变化。而包名由应用程序开发者在编写程序时确定,它在系统中是稳定不变的。同时,在程序发布时也有关于包名的验证,避免了出现重复的情况,从而保证了包名在系统中的唯一性和稳定性。

通常,Android 系统的应用程序包名为:com.android.应用程序名,如浏览器的包名为“com.android.browser”。而第三方应用程序的命名方式为“公司域名的逆序.应用程序名”,如百度输入法的包名为“com.baidu.input”。这种极具规则的命名方式非常适合用于域命名,并且便于监控和管理域。具体的域命名方式是将包名中的“.”换成“/”,并且添加“<kernel>”作为命名空间值。例如,百度输入法的包名为“com.baidu.input”,则它的域名为“<kernel>/com/baidu/input”。如果一个应用程序不仅可以触发多个进程,而且可以使进程独立,则程序会为被触发的进程赋予与触发进程相同的域名。

同时,为了防止恶意破坏者和入侵者盗用合法应用程序的包名进行注册,在生成域名之前需要对应用程序的签名进行检测,检测结果正确则执行生成域名的过程,否则拦截该应用程序。

3.2.2 域生成

根据域名生成规则,应用程序的域名并非由执行过程决定,而是由包名决定,因此应用程序的域不会随着程序的执行自动生成。这就需要实现一种机制,即可以让这些应用程序的域自动生成,并且需要在程序启动之前完成。这一功能具体分两部分来实现:1)在开机时,扫描当前系统中已经安装的应用程序,获取它们的包名,并按照域命名规则生成域名;2)在用户安装一个新应用程序时,获取它的包名,生成该应用程序的域名。这两个过程保证了每个应用程序都会获得一个属于它的域。

域生成的过程必须在应用程序启动之前完成,主要有两个原因:1)应用程序要遵循域规则来运行,这就要求系统中要先有该应用程序的域和域规则;2)只有域生成过程在应用程序启动之前完成,才能完整地记录下应用程序在运行过程中执行的所有操作。当然,在系统开机之前完成域生成会付出时间代价,但用较小的时间代价换取更加安全的运行环境是可以接受的。

3.2.3 域转换

当应用程序启动后,系统将会通过 zygote 的 forkAndSpecializeCommon 函数来执行/system/bin/app_process,从而生成该应用程序的进程。所有应用程序的启动过程都是如此。如果不做修改,所有应用程序的域名都会是<kernel>/system/bin/app_process。为了将不同的应用程序划分到各自的域中,必须在 forkAndSpecializeCommon 函数的调用过程中改变应用程序的域名,将其转到以自己的包名命名的域中,这一过程主要分为两步:1)通过 PackageManager 类获得当前应用程序的包名,并根据域命名规则得到该应用程序的域名;2)调用 PackageManager 类重新设置,将应用程序的域转到该域中。一旦成功完成域转换,应用程序在运行过程中都将遵循转换后的域规则。域转换模块保证了系统中所有由 zygote 生成的应用程序在启动时就会被划分到不同的域中。这样,管理人员就可以针对不同的域制定不同的规则,从而实现以应用程序为单位来控制整个 Android 系统。

3.2.4 域固定

根据本文安全增强机制的设计,应用程序在经过域转换过程转换到属于自己的域之后,需要固定在该域中,不能再随着执行操作而发生新域的生成和域转换。要实现这一功能,需要在策略文件中添加 keep 规则。keep 规则的具体格式如下:keep_domain any from <kernel> xxx,xxx 表示该应用程序的域名。以百度输入法为例,其规则为:keep_domain any from <kernel>/com/baidu/input。

3.3 安全策略规则的设计与制定

安全策略规则库由具体规则组成,描述了主体(安全域)对客体(各种系统资源)的权限信息。

3.3.1 安全策略规则的设计

1) 进程安全级别的设计

Android 系统中的进程可以分为系统守护进程、系统应用进程和第三方应用进程。守护进程(如 zygote, init, vold 等)应该拥有最高的安全级别。系统应用进程(如联系人、电话、短信和拍照等)维护着系统的基本功能,应该属于中等安全级别。第三方应用进程(如浏览器、输入法、音乐播放器等)应该受到最严密的监控和管理,因此其安全级别应该是最底的。

2) 文件资源划分的设计

按照用户和安全级别,系统资源主要分为系统文件、策略文件、安全文件、机密文件、数据文件和普通文件。其中,系统文件、策略文件和安全文件属于系统的核心资源,应该受到最严密的保护;机密文件应该只属于有权访问它们的进程;数据文件是分属于每个应用程序的资源文件,理论上每个进程都不能访问其他进程的数据文件,但对于一些特殊的系统应用,可以出现共享的情况;普通文件的安全级别最低,可以设置为共享文件。

3) 多级别访问规则的设计

系统中的进程被划分到某一个安全级别,通过对安全级别设置访问权限,增强系统的安全性。守护进程的安全级别最高,可以访问多数资源(如系统文件、机密文件等)。系统应

用程序除了有权访问系统文件和普通文件外,还可以访问属于自己应用功能范围内的数据文件,同时还可以访问与之相关的其他进程的数据文件。第三方应用程序只可以访问部分必需的系统文件和属于自己进程的数据文件。

4) 域独有访问规则的设计

系统中的进程,除了可以获得所属安全级别的访问权限外,还可能需要一些独有的访问权限,将这些权限单独赋予该域,并写入该域的访问规则中。如果将原有安全机制赋予进程的权限称为集合 M ,应用程序安全级别赋予的权限称为集合 A ,单独赋予应用程序的权限称为集合 B ,应用程序真正拥有的权限称为集合 K ,那么 $K=A \cup B \cap M$ 。

3.3.2 安全策略规则的制定

安全策略的制定,首先需要通过学习过程了解系统中各个进程在正常运行过程中执行的操作,得出这些数据后对其进行分析,再根据分析结果结合需求制定策略。

1) 策略学习

策略学习过程中,系统会监视进程的所有操作,并对其记录和分析,主要分析系统环境变量、读写的数据文件、软硬件加速、日志、内存池和执行操作等信息。

2) 默认规则的制定

第三方应用程序在运行过程中执行的操作和安全防护要求类似,因此需要为第三方应用程序制定默认的规则。规则的内容主要包括 3 个部分:①profile 决定了需要监控和审核的操作,因此需要制定新的 profile;②制定默认许可,包括访问 OTHER_PATHNAME 包含的资源、以非 root 的身份进行执行操作、执行除 /system/bin/su 外的命令;③为第三方应用程序的域制定 use_profile 和 use_group,这样它们就会遵循制定的 profile 和 group 的规则。

3) 文件资源划分规则的制定

系统中的资源可以被划分为 4 部分:POLICY_PATHNAME, TOMOTYO_PATHNAME, SEC_PATHNAME 和 OTHER_PATHNAME。按照应用程序被授予的不同级别权限,分类获取文件资源。

4) 多安全级别规则的制定

可以使用 use_group 对不同安全级别的应用程序进行分类,与默认规则制定的方法类似,同样依靠 profile 来确定操作的审核范围。

5) 域独有访问规则的制定

制定独有访问规则主要与 mount 有关。需要执行 mount 操作的域有 4 个,最重要的就是 <kernel> /system/bin/app_process,因为它与第三方应用程序紧密相关,因此需要严格监控。首先需要通过学习得出该域必需的许可,然后将这些许可添加到该域的规则中。

本文主要提出了安全策略规则制定的思想,由于篇幅限制,本文不再具体描述。

4 实验验证

本节对改进了域架构的系统安全增强机制进行实验验证,主要包括可行性验证、兼容性验证和有效性验证。

4.1 实验 1:安全域生成实验

本节主要验证改进后的安全域算法是否有效,主要对域生成、域转换及域固定进行验证。

以来百度输入法、短信管理程序、通话管理程序等应用程序为例,来检验改进后的安全域功能是否有效。由于在学习模式下,域策略文件(domain_policy.conf)会详细记录进程的运行状态,因此观察并分析该文件的内容,可以判断域管理功能是否有效。

1) 实验过程

①设置运行模式为学习模式(Learning 模式),即使用 use_profile 文件中模式为“learning”的配置,该模式的编号为 1。

②增加安全域固定策略,防止域转化的发生,即在策略文件中加入规则:keep_domain any from <kernel> /com/baidu/input。

③下载并安装百度输入法,其包名为 com.baidu.input。

④重启系统后,域策略文件(domain_policy.conf)会实时记录当前手机系统中各进程的运行状态,每个应用程序的操作都会被如实记录下来。

⑤查看域策略文件(doman_policy.conf),从而可以掌握系统中各个应用程序的运行过程。

2) 实验结果及分析

图 3 是策略文件的一个部分。由于每个域的行为记录都较长,为方便说明,图中省略了大部分的行为记录。

```
<kernel> /com/baidu/input
use_profile 1
use_group 0
file read /data/data/com.baidu.input/files/loginshare.json
file read /data/data/com.baidu.input/files/log
...
<kernel> /com/android/phone
use_profile 1
use_group 0
file read /write /dev/binder
file read /data/data/com.android.providers.telephony/shared_prefs/preferred-apn.xml
...
<kernel> /com/android/smspush
use_profile 1
use_group 0
network unix stream connect \000jdpw-control
file read/write /dev/binder
file write debugfs:/tracing/trace_marker
file read/write /dev/ashmem
...
<kernel> /com/ancode/anrom
use_profile 1
use_group 0
network unix stream connect \000jdpw-control
file read/write /dev/binder
...
```

图 3 部分域的行为记录

Fig. 3 Behavior record of partial domain

如图 3 所示,包名为 com.baidu.input 的百度输入法生成

的安全域为 $\langle \text{kernel} \rangle / \text{com}/\text{baidu}/\text{input}$, 它执行的所有操作都被记录下来, 如读取了 `loginshare.json`, `log` 等文件; 系统中的其他应用程序同样成功地根据包名划分开, 每一个应用程序的进程归属于一个域, 应用程序运行中的行为已经被记录下来, 如系统应用程序 `com.android.phone` 生成的安全域为 $\langle \text{kernel} \rangle / \text{com}/\text{android}/\text{phone}$ 等。以上说明, 改进后的域管理模块解决了原始域名生成的问题, 同时由于安全域名为 $\langle \text{kernel} \rangle / \text{com}/\text{baidu}/\text{input}$, 而非 $\langle \text{kernel} \rangle / \text{system}/\text{bin}/\text{app_process}/\text{com}/\text{baidu}/\text{input}$, 并且未生成或转换到其他安全域名, 因此说明域固定策略发挥了作用。

4.2 实验 2: 改进域架构机制的兼容性验证

选取 Android 针对 ARM 架构的各类版本, 分别以内核补丁的方式应用改进的域架构机制来进行验证。

1) 实验过程

①准备 Android 操作系统。分别从官网下载 9 个版本的原生操作系统, 分别是 2.3.6, 3.0, 4.0, 4.1.2, 4.2.2, 4.3, 4.4, 5.0 32bit 及 5.0 64bit。

②依次将改进后的域架构应用到上述系统中。

③验证是否可以正常运行。

2) 实验结果及分析

具体实验结果如表 1 所列。从表中可以看出, 改进的域架构对大部分的 Android 版本都有较好的兼容性, 但是在应用到 5.0 时, 没有成功, 原因在于测试的 5.0 是 64 位版本, 而本文改进的域架构是针对 32 位开发的, 需要处理兼容问题, 而针对 5.0 版 32 位的测试结果是成功的, 从而印证了上面的判断。另外, 由于 4.2 之后的 Android 系统自身已经应用了 SEAndroid 安全增强机制, 说明本文提出的安全增强机制可以和 SEAndroid 等安全增强机制并存。

表 1 改进的域架构的兼容性实验结果

Table 1 Compatible results of improved domain architecture

Android 版本	应用结果	运行结果
Android OS 2.3.6	OK	OK
Android OS 3.0	OK	OK
Android OS 4.0	OK	OK
Android OS 4.1.2	OK	OK
Android OS 4.2.2	OK	OK
Android OS 4.3	OK	OK
Android OS 4.4	OK	OK
Android OS 5.0 64-bit	NO	—
Android OS 5.0 32-bit	OK	OK

4.3 实验 3: 有效性验证

李凡^[17]通过修改 Android 系统中的 SQLite 源代码, 替换了原有的 `libsqlite.so` 库, 实现了可加密的 SQLite 库, 为系统中的数据库安全提供了保证(以下将其简称为 SecEnfore)。本文提到的方案(以下简称为 EnforeDroid)可以通过在策略文件中加入相应的策略规则, 来达到使敏感目录下的文件不被其他进程读取的目的, 同样起到了数据保护的作用。本实验以多个敏感目录为测试对象, 如 `/sys/kernel/security` (即 `securityfs:/`), `/sys/kernel/security/tomoyo/`, `/data/sec/`, `/system/bin/` 等, 通过在策略文件中加入相应的策略规则, 来达到

使该目录不被测试进程读取的目的。

1) 实验过程

①制定保护策略, 设置第三方应用程序和 adb shell 都无法查看的测试目录, 具体的安全策略规则见 3.3.2 节。

②将 EnforeDroid 的运行模式设置为强制模式。

③启动安卓系统中的文件浏览器软件文件 Root Explorer 来查看测试目录, 检查是否可以浏览目录。

④通过 adb 连接到安卓 shell, 在 shell 下使用 `ls` 命令, 测试是否可以浏览目录。

⑤查看 EnforeDroid 的日志文件, 查找审计记录。

⑥设置运行模式为禁止模式, 重复步骤③—步骤⑤。

2) 实验结果及分析

在强制模式下, 图 4 显示了 Root Explorer 无法浏览测试目录下的文件; 图 5 显示了在 adb shell 模式下, 其也不能查看测试目录下的文件; 图 6 给出了在这个过程中产生的日志文件内容。通过分析发现, 黑框和下划线部分验证了上述操作的结果。在禁用模式下, Root Explorer 及 adb shell 均可以访问测试目录。



图 4 在强制模式下 Root Explorer 查看测试目录

Fig. 4 Viewing test directory using Root Explorer on forced mode

```
shell@hammerhead:/ $ ls -al /sys/kernel/security/
opendir failed, Permission denied
```

图 5 在强制模式下命令行查看测试目录

Fig. 5 Viewing test directory using CLI on forced mode

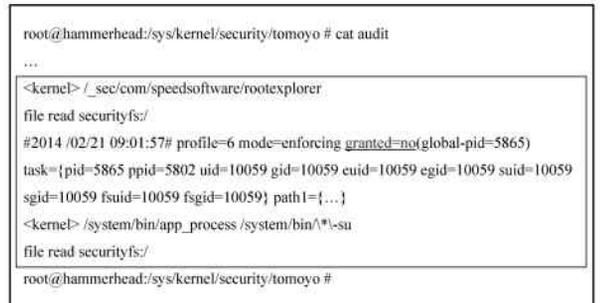


图 6 安全日志

Fig. 6 Security log

在本实验中, 非授权的进程 (adb shell) 无法查看机密文

件,第三方应用程序(Root Explorer)甚至无法觉察到机密文件的存在。由此可见,EnforeDroid能够保护敏感目录,防止被未授权的进程和第三方应用程序访问,可以保护敏感资源的安全。

文献[17]证明了SecEnfore通过加密可以保护数据库的安全,而本实验则证明了EnforeDroid可以通过设定策略保护敏感目录下的任何数据文件的安全,相比之下,EnforeDroid的应用范围更广。

4.4 实验结论

通过实验可以看到,安全域功能可有效运行。安全域是TOMOYO Linux实现强制访问控制(MAC)的基础,改进后的域名生成设计更加切合安卓系统的特点,通过分析域策略文件的内容来说明安全域功能可以有效运行,且可兼容大部分安卓系统。SELinux安全增强机制覆盖的安卓版本比较少(4.2之后的版本),而本文提出的安全增强机制可以覆盖大部分安卓版本(从2.x到5.x)。最后,通过对比实验,验证了EnforeDroid可以有效保护敏感资源的安全,与SecEnfore相比,其应用范围更广。

结束语 本文根据Android与Linux系统的差异性,汲取了Linux系统中强制访问控制的理论思想和实现方法,结合Android系统的特性,基于TOMOYO Linux域架构,设计并实现了适合Android系统的安全增强机制。该机制实现了在内核层控制系统中应用程序的操作,将系统控制深入到进程级别,并针对不同进程设置不同权限的精细控制,从而改进了安全控制机制,细化了控制粒度,提升了Android系统的安全防护能力。在保证Android系统正常运行的基础上,将系统中的应用程序的权限控制在最小范围内。

参 考 文 献

- [1] SHABTAI A, FLEDEL Y, KANONOV U, et al. Google Android: A Comprehensive Security Assessment[J]. IEEE Security & Privacy, 2010, 8(2): 35-44.
- [2] BHASU A. Security Enhanced Linuses LIDS, TOMOYO and SELINUX[R/OL]. [2011-12-15]. http://it.toolbox.com/blogs/linux-security/security-enhanced-linuses-lids-tomoyo-and-se_linux-4977.
- [3] LIU J, SUN K Q, WANG S L. Vulnerability analysis of the Android operating system code based on control flow mining [J]. Journal of Tsinghua University (Science and Technology), 2012, 52(10): 1335-1339. (in Chinese)
刘剑, 孙可钦, 汪孙律. 基于控制流挖掘的Android系统代码漏洞分析[J]. 清华大学学报(自然科学版), 2012, 52(10): 1335-1339.
- [4] COKER R. Porting nsa security enhanced linux to handheld devices: Proc of Linux symp[C]//Ottawa: Linux symp Inc. 2003: 117.
- [5] SHABTAI A, FLEDEL Y, ELOVICI Y. Securing Android powered mobile devices using SELinux[J]. Security & Privacy, 2010, 8(3): 36-44.
- [6] BUGIEL S, DAVI L, DMITRIENKO A, et al. Practical and lightweight domain isolation on android[C]//Proc of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. New York: ACM, 2011: 51-62.
- [7] SMANEY S, CRAIG R. Security enhanced (SE) Android: bringing flexible MAC to Android[C]//Proc of the 20th Annual Network and Distributed System Security Symp. USA: ANDSSS, 2013: 1-5.
- [8] BUGIEL S, HEUSER S, SADEGHI A R. Flexible and fine-grained mandatory access control on Android for diverse security and privacy policies[C]//USENIX Association. Berkeley: USENIX Association, 2013: 131-146.
- [9] DING M, YU L. Design and Implementation of Android OS Security Enhancement Scheme [J]. Computer Knowledge and Technology, 2015, 11(33): 19-20. (in Chinese)
丁汨, 于琳. Android系统安全增强方案的设计与实现[J]. 电脑知识与技术, 2015, 11(33): 19-20.
- [10] LU Y M, LI Y F, YING L Y, et al. Security Analysis and Enhancement of Third-Party Android Push Service[J]. Journal of Computer Research and Development, 2016, 53(11): 2431-2445. (in Chinese)
路晔绵, 李轶夫, 应凌云, 等. Android应用第三方推送服务安全分析与安全增强[J]. 计算机研究与发展, 2016, 53(11): 2431-2445.
- [11] WAN Y, WANG G L, FENG X Y. An evaluation model for information security of Android application based on analytic hierarchy process[C]//World Automation Congress. USA: WAC, 2016: 1-6.
- [12] XU M Z, WUN W Q, ALAM M. Security enhancement of secure USB debugging in Android system[C]//Consumer Communication and Networking Conference (CCNC). USA: IEEE, 2016: 134-139.
- [13] Mergemstatus[EB/OL]. <http://seandroid.bitbucket.org/Merge-Status.html>.
- [14] ELECTRONICS S. Samsung KNOX[EB/OL]. <http://www.samsung.com/global/business/mobile/solution/security/samsung-knox>.
- [15] Knox workspace supported devices[EB/OL]. <https://www.samsungknox.cn/en/solutions/knox/technical/knoxsupporteddevices>.
- [16] TOMOYO Linux[EB/OL]. <http://tomoyo.sourceforge.jp>.
- [17] LI F. Analysis and Enhancement of Security Mechanism Based on Android OS[D]. Wuhan: Huazhong University of Science & Technology, 2012. (in Chinese)
李凡. Android系统安全机制的分析与增强[D]. 武汉: 华中科技大学, 2012.