

基于 gdb 的 Android 软件漏洞挖掘系统

杨海民,张 涛,赵 敏,鲁小杰

(解放军理工大学 指挥信息系统学院,江苏 南京 210007)

摘 要:传统的漏洞挖掘技术一般适用于 x86 平台,且是面向 PC 的。随着 Android 手机的普及,需要有其上运行软件的漏洞挖掘技术。针对当前 Android 软件市场审核宽松以及该领域研究相对较少等方面存在的一些问题,设计并实现了一种基于 gdb 的 Android 软件漏洞挖掘系统。系统采用基于信息流追踪的污点分析技术,从污点标记、污点传播和污点检测三个方面进行设计,并通过指令模拟执行提高分析覆盖率。当系统发现可疑漏洞时,把结果通知给用户,并能对漏洞做出全面的分析。通过对 Android 软件的测试,发现了部分软件中的缓冲区溢出漏洞,证实了系统的有效性。

关键词:Android 软件;漏洞挖掘;污点分析;gdb

中图分类号:TP302.1

文献标识码:A

文章编号:1673-629X(2015)08-0156-05

doi:10.3969/j.issn.1673-629X.2015.08.033

Android Software Vulnerabilities Mining System Based on gdb

YANG Hai-min,ZHANG Tao,ZHAO Min,LU Xiao-jie

(Institute of Command Information System,PLA University of Science and Technology,Nanjing 210007,China)

Abstract:The traditional vulnerability mining techniques are generally applicable to x86 platform,and intent to the PC. With the popularity of Android phones,the vulnerability mining technology running on it is needed. Because of the problems of the current accommodative Android software market audit and relatively small research in this area,a gdb-based Android software vulnerabilities mining system is designed and implemented. The system adopts taint analysis techniques based on tracking the flow of information,which is designed from taint marking,taint transmission and taint detection,and improves analysis coverage through instruction simulation. The result is notified to the user and the system can make a comprehensive analysis of vulnerability when the system finds the suspicious loopholes. By the test for Android software,some buffer overflow vulnerabilities in software is found,which proves the effectiveness of the system.

Key words:Android software;vulnerabilities mining;taint analysis;gdb

1 概 述

随着 Android 系统在全球市场份额的不断增大,Android 手机用户越来越多,同时 Android 软件应用也越来越广泛。来自思科公司的 2013 年度安全报告^[1]显示,Android 设备已经成为大多数恶意软件攻击的主要目标,大约有 71% 的 Android 用户遭到了攻击。国内用户通过各种第三方渠道安装 Android 应用,一方面由于市场审核一直较为宽松,另一方面由于用户在安装应用时对安全关注不够,导致手机私密信息泄露事件频繁发生。

当前 Android 终端受到的攻击主要来源于通过包装或者修改等手段形成的恶意软件,这些软件以非法手段获取用户的重要隐私信息。但随着 Android 软件

中音频、视频软件解码器的编写涉及到 CPU 的高级性能运算,以及需要运行其他平台上 C、C++语言编写的游戏,Android 软件中包含越来越多的 C、C++编写的.so 动态库。Android 终端每年都会受到频繁的攻击,攻击者会针对 Android 软件开发方式的转变,采取有针对性的方式进行攻击。因此,随着 Android 软件中包含 C/C++代码比例的增大以及软件数量的增多,C/C++语言存在的缓冲区溢出漏洞问题不可避免地将会成为 Android 手机潜在的威胁。

文中针对 Android 软件中存在的缓冲区溢出问题,设计并实现了一种基于 gdb(GNU Debugger)的 Android 软件漏洞挖掘系统。该系统主要采用污点分析技术,从污点标记、污点传播和污点检测三个方面针对

收稿日期:2014-09-16

修回日期:2014-12-19

网络出版时间:2015-07-21

基金项目:国家科技重大专项基金资助项目(2012ZX03006-003)

作者简介:杨海民(1990-),男,硕士生,CCF 会员,研究方向为嵌入式系统应用及信息安全;张 涛,教授,研究方向为嵌入式系统应用及信息安全。

网络出版地址:<http://www.cnki.net/kcms/detail/61.1450.TP.20150721.1439.022.html>

漏洞挖掘进行设计。首先对从外部引入的数据进行添加污点标记处理,在此基础上对程序运行的指令进行具体分析,修改相关数据的污点标记,跟踪并监视外部数据传入应用后的使用情况,最后当外部数据由于被非法使用导致缓冲区溢出时,系统检测到漏洞并把详细情况呈现给用户。

2 相关工作

2.1 Android 系统架构

Android 分为五层,从高层到底层分别是应用程序层(Applications)、应用程序框架层(Application Framework)、系统运行库层(Libraries 和 Android Runtime)和 Linux 内核层(Linux Kernel)。

应用程序层:Android 一般会与核心应用程序包一起发布,该应用程序包包括主屏(Home)、E-mail 客户端、SMS/MMS 短信息程序、日历、地图、浏览器、联系人管理程序等。

应用程序框架层:为开发人员提供了可以安全访问核心应用程序所使用的 API 框架。

系统运行库层:系统运行库层包括程序库和 Android 运行库两部分。程序库包含一些 C/C++ 库,这些库能被 Android 系统中的不同组件使用,它们通过应用程序框架为开发者提供服务。Android 运行库又分为核心库和 Dalvik 虚拟机两部分。核心库提供了 Java 语言库的大多数功能,这里主要通过 JNI 的方式向应用程序框架层提供调用底层程序库的接口。Dalvik 虚拟机是为了能同时高效地运行多个 VMs 而实现的。

Linux 内核层:Android 依赖于 Linux 2.6 版内核提供的核心系统服务,例如安全、内存管理、进程管理、网络栈、驱动模块等^[2]。

Android 应用程序开发是 Android 开发的一个重要方面,Android 应用程序大部分是使用 Java 语言编写的,通过调用应用程序框架层所提供的 API 来完成。随着 Android 平台下软件应用功能需求的增多,仅使用 Android SDK 通过 Java 语言编写程序已经不能满足开发者了。如图 1 所示,目前应用比较广泛的是使用 Java 通过 JNI 的方式配合 Android NDK 来开发原生应用程序,提高了应用程序的效率。

2.2 gdb

gdb^[3] 是 GNU 开源组织发布的一个强大的 Unix 下的程序调试工具,通过设置断点命令控制被调试的程序在所指定设置的断点处停止运行,当程序被中断时,可以检查此时程序中变量的值和地址以及寄存器和内存的状态信息。

2.3 污点分析研究现状

污点分析是指对非信任来源的数据进行标记,并

追踪其在程序执行中的传播过程,以达到获取关键位置与输入数据关联信息的分析方法^[4]。污点分析有 3 个主要组件:taint sour(标记输入);taint propagation(传播污点数据)和 taint sink(污点信息终点)^[5]。目前,国内外研究机构均围绕此开发了许多分析工具,较著名的有:TaintCheck^[6]、Dytan^[7]、Argos^[8]。

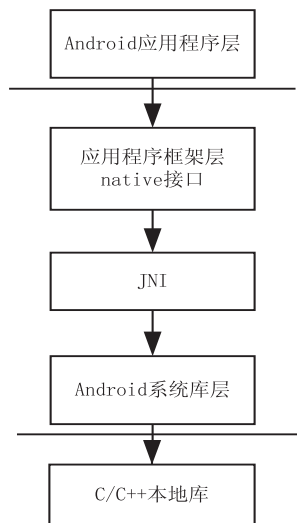


图 1 JNI 层次结构图

TaintCheck、Dytan 和 Argos 都是针对 x86 指令集且面向 PC 的,它们建立在不同的 x86 模拟器上,不适用于文中研究的面向手机终端的针对 ARM 指令集的漏洞检测。基于动态污点的 TaintCheck 分析可以检测受控执行、格式化字符串、污染系统调用等漏洞且误报率低。TaintCheck 对不可信来源的输入标记,针对每个字节存储单元有 4 字节的影子内存来存放一个指针,这个指针指向一个 Taint data Structure 数据结构来记录污点相关信息,在污点传播模块其动态地将执行指令翻译成 UCode 形式的中间表示后,跟踪每条指令决定结果是否被污染。UCode 指令分成 3 类,数据移动指令、算术指令及其他 NOP、JMP 等指令。在污点检测方面,TaintCheck 把污点数据作为跳转地址、格式化字符串和系统调用参数认为是危险操作,另外还会针对一些特定应用和库的攻击做检查。Dytan 主要增加了对隐式操作的分析,增加了污点分析的全面性。Argos 和 TaintCheck 原理上差不多,但其考虑到了控制流也就可以增加对隐式操作的分析,可有效检测缓冲区溢出和格式化字符串漏洞。Argos 的缺点也是显而易见的:低效,比起直接运行在实际的主机上的程序,运行在 Argos 上的程序速度会减慢 10~30 倍。文中在借鉴 TaintCheck 污点源标记技术的基础上,没有采用指令翻译,而是针对源操作数不同的污染状况采用指令预处理来模拟指令处理,可对隐式操作进行有效分析,避免了指令翻译过程中可能存在的误翻和漏翻问题。

针对 Android 平台的信息流追踪框架有 Taint-

Droid^[9],其主要基于应用级的污染追踪,在相关的 API 调用处把隐私数据标记为污染数据,然后使用多层跟踪追踪污点数据在 Dalvik 虚拟机中的传播,当污点数据在进程通信中被使用,修改过的 binder 库保证污点标记包含所有相关的数据,最终污点标记被打包通过内核传播后被远程不可信应用接收。假如不可信应用将污点标记的私密信息泄露出去,系统就会产生一个报告。显然 TaintDroid 只是防止内部私密信息不被泄露,对于代码中存在的漏洞并没有检测能力,另外其基于应用程序层,对于纯 C 的应用或者通过 native 代码实现功能的应用无能为力。

3 基于 gdb 的漏洞挖掘技术

文中设计并实现了一种基于 gdb 的 Android 软件漏洞挖掘系统。在污点分析过程中,首先给程序外部来源数据添加污点标记,通过对程序运行中的指令进行分析修改数据的污点标记,在此基础上通过检测数据的不合理利用发现软件存在的漏洞。通过修改 gdb 源代码实现了原型系统,图 2 是系统的组成架构。其主要思想是在 gdb 源代码基础上,针对 Android 平台和污点分析需要做出修改,添加了污点标记模块、污点传播分析模块和污点检测模块,重新编译得到相应的 gdb 和 gdbserver。gdb 和 gdbserver 同时运行在 Android 手机终端,方便检测漏洞。

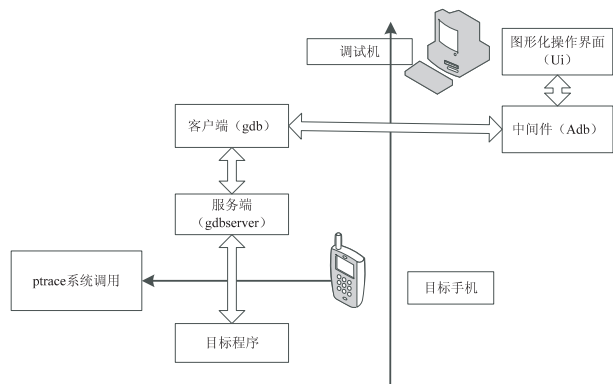


图 2 Android 软件漏洞挖掘系统组成架构图

利用 gdb 可以根据函数名称和指定条件设置断点,根据污点标记、污点分析和污点检测的原则,把相应模块加入到设定的在 gdb 源码目录基础上添加污染源函数断点处理、指令预分析处理和污染检测断点处理模块。根据 gdb 源码中已有的断点定义和断点设置,通过断点链表管理定义的关于污染源函数和污染检测断点,指令预分析模块是指在指令真正执行之前,模拟执行每条指令,并设置下一条指令执行地址处为断点。

3.1 污点标记

污点标记是污点传播的首要步骤,其标记方式极

大地影响着污点传播处理与污点信息存储的效率^[10]。污点标记中最主要的工作是对污点数据的标记进行添加和删除工作。针对 Android 软件,在获取所有寄存器上下文的基础上,通过污染检查函数来设置断点,污染检查函数通过判断当前 pc 是否是 B/BL 函数调用指令,再做进一步的具体判断。判断标准是根据函数名,污染源主要来源于文件输入和打开函数,给这些函数引入的数据添加污点标记,数据状态分为 tainted 和 untainted,同时定义 struct TAINTMEM 来记录污染地址。

```
struct TAINTMEM {  
    unsigned int StartAddress;  
    unsigned int AddressLength;  
}
```

对于内存污染列表结构的定义与实现,可以通过页表实现。针对 ARM 微处理器共有 37 个 32 位寄存器,其中 31 个通用寄存器,6 个状态寄存器,定义数组来对寄存器污染列表结构进行定义,数组中每个元素结构为 struct TAINTRREG。

```
struct TAINTRREG {  
    char val; //8bit/16bit/32bit/64bit  
    unsigned int src[4]; //污染源  
}
```

ARM 指令大多数是有条件执行的,负标志 N、零标志 Z、进位标志 C、溢出标志 V 的内容都可被算术或逻辑运算的结果所改变,标志位的状态存储于 1 字节大小的固定内存中。

在污点标记模块主要完成两方面的工作:污点添加和污点删除,即 untainted→tainted 和 tainted→untainted 操作。

(1)在污点添加模块,主要针对 open、fopen、fgets、mmap、read、fread、scanf、malloc、realloc、recv、recvfrom、getcwd 函数做具体处理。把上面的函数分为两类,第一类是 open 和 fopen 函数,记录打开的文件名,针对文件名的索引标识所做操作的地址添加到污染源列表;第二类是 fgets、mmap、read 等函数,此类函数的处理则是通过栈读取获取参数直接添加到污染源列表。

(2)在污点删除模块主要针对两方面情况:一方面是 free 函数,它将分配的内存返回给操作系统进行回收,将该内存对应的污点状态删除;另一方面是污点被常量或者没有污染的数据赋值,这方面主要结合污点传播中的指令预分析。

3.2 污点传播

ARM 指令主要分为 6 种,包括跳转指令、数据处理指令、程序状态寄存器传输指令、Load/Store 指令、协处理器指令和异常中断产生指令^[11]。图 3 是 ARM 指令的一般编码格式。

31-28	27-25	24-21	20	19-16	15-12	11-0
Cond	001	opcode	s	Rn	Rd	Shifter_operand

图 3 ARM 指令的一般编码格式

其中,Cond 为条件,001 为固定编码,opcode 为操作符,s 为影响标志,Rn 为第一操作数编码,Rd 为目的寄存器,Shifter_operand 为第二操作数。

由于 ARM 指令集是 Load/Store 型的^[12],把指令重新分为三类:

(1)影响内存到寄存器污点传播的即加载指令,其用于将存储器中的数据传送到寄存器,包括 LDR、LDRB 和 LDRH。

(2)影响寄存器到内存传播的即存储指令,其用于完成与加载指令相反的工作,包括 STR、STRB 和 STRH。

(3)影响寄存器到寄存器污点传播的,包括除了加载指令和存储指令之外的其他指令。

污点传播模型建立在序偶的基础上,数学上由两个元素 x,y 构成的有序组,称为一个序偶,记为 $\langle x,y \rangle$ 。设 A,B 为集合,将 A 中的元素作为第 1 元素, B 中的元素作为第 2 元素,定义 $A \times B = \langle x,y \rangle$,其中 $x \in A,y \in B$ 。

根据上面定义, $\langle SA,A \rangle$ 为污点传播中的污点数据,约定 SA 为污点引入的内存地址数据的集合, A 为被污点传播感染的地址数据的集合,那么 $\langle sa,a \rangle$ 为污点传播中的污点数据就包含了污染源信息,其中 $sa \in SA,a \in A$ 。

污点传播采用粗细结合的分析方法。在污点传播中,如果一条指令中的源操作数用 sn 表示,目的操作数用 dn 表示,那么 $sa \in A \rightarrow a \in A$,也就是传播中源是污染的,就定义目的污染,无论源和目的是内存还是寄存器。另外,定义 $\langle sa,a_1 \rangle \times \langle a_1,a_2 \rangle = \langle sa,a_2 \rangle$,这样方便找到真正的污染源。

关于标志位状态的改变需要结合具体指令,文中对每条 ARM 指令进行了预处理,模拟处理每个操作数和状态位分别在 tainted 和 untainted 状态下,目的操作数的状态改变情况。假设 ARM 指令的数量为 n ,源操作数和状态位的数量总和是 s ,那么预处理的指令数量将会达到 $n * 2^s$ 。在每条指令真正执行之前根据指令名称和操作数以及状态位污染情况需要与预处理指令进行匹配。

3.3 污点检测

污点检测过程中把断点设置在容易引发漏洞的相应位置进行污点检测,检测使用污点数据的不安全方式^[13],主要通过这些位置添加污点检测模块。针对漏洞可能存在的不同形式,文中采用粗细结合的污点

检测方法。一方面确定一些危险函数调用作为检测对象,这些函数主要包括 strcpy、strcat 类型函数^[14],当这些函数的源字符串数据具有污点标记时,很可能引发缓冲区溢出漏洞,检测过程如图 4 所示。

另一方面结合指令预处理,针对有些语句实现的功能和上面需要检测的函数相同,但这些语句经过上面的检测过程不会被发现。因此,在检测过程中利用指令与固定模式指令匹配,当匹配成功时则认为该条指令引发漏洞。定义当指令为数据传送指令 MOV 和 MVN 且第一操作数具有污点标记时,则认为这是可能导致缓冲区溢出漏洞的指令。

4 实验

按照上面的设计,文中实现了 Android 智能终端软件安全漏洞检测系统,并对该系统进行了相关功能测试。表 1 是实验需要的软硬件环境。

表 1 实验软硬件配置

名称	数量	配置
Android 移动设备	1	推荐内存 1 G 以上、SD 卡内存 2 G 以上,Android OS 版本 4.0.4 以上,拥有 root 权限
USB 数据线	1	
PC 机	1	推荐内存 2 G 以上,硬盘 40 G 以上,OS 推荐 XP SP3 以上版本

由于目前系统检测效率还在优化,实验主要针对 Android 软件市场上 10 个代码数据量较小的应用做了检测。发现在检测一款记事本和 MP3 音乐播放器时发现了缓冲区溢出漏洞,并根据污染源信息修改输入数据长度导致了程序的崩溃,证明了检测方法的有效性和实用性。其中 MP3 音乐播放器存在打开 .list 畸形文件导致的缓冲区溢出漏洞。该漏洞形成过程主要通过 fopen 函数打开一个 .list 格式文件,然后通过 fgets 函数读取文件内容,经过一系列污点传播,最后由于调用 strcpy 函数引发漏洞。

5 结束语

文中在污点分析技术的基础上,在 gdb 源码的基础上增加了污点源函数处理、指令预分析和污染目的函数处理模块。在指令预分析模块利用对每条 ARM 指令的模拟处理,分析污点的传播,利用数学上序偶的概念来表示污点信息并进行存储,有利于得到污染源信息,在此基础上实现了 Android 软件漏洞挖掘系统。通过检测实验,发现该系统在检测速度上还有待提高,另外误报率需要进一步降低,结合这两点进一步完善系统功能,这将是下一步的工作。

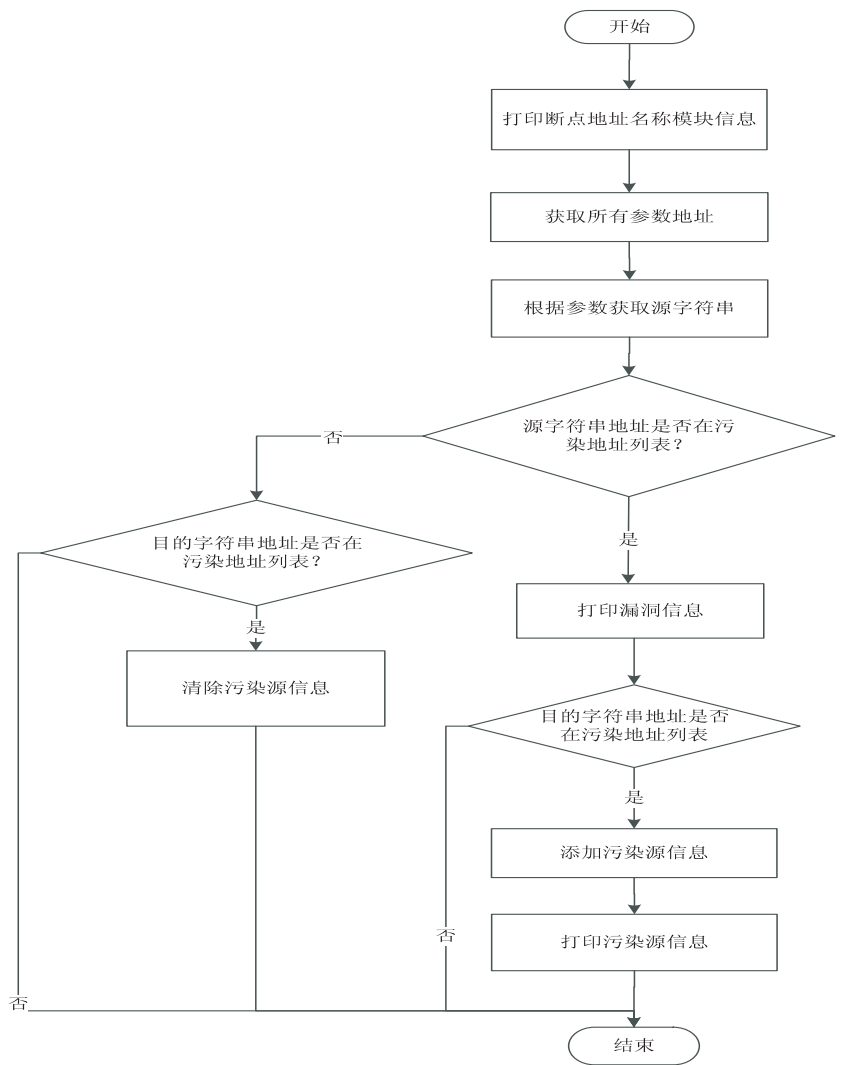


图 4 污染目的函数检测流程图

参考文献:

[1] Cisco 2013 annual security report[R/OL]. [2013-06-01]. <http://www.cisco.com/>.

[2] 杨丰盛. Android 技术内幕·系统卷[M]. 北京:机械工业出版社,2011.

[3] Gdbmanual[EB/OL]. [2014-04-04]. <http://baike.baidu.com/view/639266.htm>.

[4] Lam M S, Martin M C, Livshits V B, et al. Securing Web applications with static and dynamic information flow tracking[C]//Proc of the 2008 ACM SIGPLAN symposium on partial evaluation and semantics-based program manipulation. New York:ACM Press,2008.

[5] 刘智,张小松.一种基于污点分析的文件型软件漏洞发现方法[J].小型微型计算机系统,2012,33(1):42-48.

[6] Newsome J, Song D. Dynamic taint analysis: automatic detection, analysis, and signature generation of exploit attacks on commodity software[C]//Proceedings of the network and distributed systems security symposium. San Diego, CA:[s. n.], 2005.

[7] Clause J, Li Wanchun, Orso A. Dytan: a generic dynamic taint analysis framework[C]//Proceedings of the international symposium on software testing and analysis. [s. l.]:[s. n.], 2007.

[8] Portokalidis G, Slowinska A, Bos H. Argos: an emulator for fingerprinting zero-day attacks[C]//Proc of EuroSys'06. Leuven, Belgium:[s. n.], 2006.

[9] Enck W, Gilbert P, Chun B G, et al. TaintDroid: an information-flow tracking system for realtime privacy monitoring monitoring on smartphones[C]//Proceedings of the 9th USENIX conference on operating systems design and implementation. [s. l.]:[s. n.], 2010:393-407.

[10] 史大伟,袁天伟.一种粗细粒度结合的动态污点分析方法[J].计算机工程,2014,40(3):12-17.

[11] 杜春雷. ARM 体系结构与编程[M]. 北京:清华大学出版社,2003.

[12] ARM Ltd. ARM926EJ-S technical reference manual[EB/OL]. [2008-06-01]. <http://www.arm.com/>.

[13] 孔德光,郑焱,帅建梅,等.基于污点分析的源代码脆弱性检测技术[J].小型微型计算机系统,2009,30(1):78-82.

[14] 黄玉文,刘春英,李肖坚.基于可执行文件的缓冲区溢出检测模型[J].计算机工程,2010,36(2):130-131.

基于gdb的Android软件漏洞挖掘系统

作者：[杨海民](#)，[张涛](#)，[赵敏](#)，[鲁小杰](#)，[YANG Hai-min](#)，[ZHANG Tao](#)，[ZHAO Min](#)，[LU Xiao-jie](#)

作者单位：[解放军理工大学 指挥信息系统学院, 江苏 南京, 210007](#)

刊名：[计算机技术与发展](#)

英文刊名：[Computer Technology and Development](#)

年，卷(期)：2015(8)

引用本文格式：[杨海民](#).[张涛](#).[赵敏](#).[鲁小杰](#).[YANG Hai-min](#).[ZHANG Tao](#).[ZHAO Min](#).[LU Xiao-jie](#) [基于gdb的Android软件漏洞挖掘系统](#)[期刊论文]-[计算机技术与发展](#) 2015(8)