基于 ARM 平台的 ROP 攻击及防御技术 *

钱 逸，王轶骏，薛 质
(上海交通大学信息安全学院，上海 200240)

[摘　要] 随着智能手机领域的发展，几乎所有智能手机及平板电脑都采用了 ARM 架构，在此平台上的安全问题也越来越受到研究者的关注。X86 平台上流行的返回导向编程被引入到了 ARM 平台上。通过研究总结 X86 平台上返回导向编程的攻击和防御机制，给出了该攻击移植到 ARM 平台上的技术细节，包括具体实现方式和 gadget 搜索算法的差异性，通过自动构建 gadgets 链加速 Exploit 开发，最后提出了一种系统库沙盒技术来防御此攻击。

[关键词] 返回导向编程；ARM 架构；gadget 搜索；系统库沙盒
[中图分类号] TP309　　[文献标识码] A　　[文章编号] 1009-8054(2012)10-075-03

ROP Attack and Defense Technology based on ARM

QIAN Yi，WANG Yi-jun，XUE Zhi
(School of Information Security，Shanghai Jiaotong University，Shanghai 200240，China)

[Abstract] With the development of mobile-phone field，ARM architecture is usually used in the mobile-phone and tablet computer，and the security issues on ARM platform always attract much attention from the researchers. ROP，a popular attack method on X86 is planted into ARM. This paper analyzes the different ROP defense methods on X86，presents the technical details of ROP attack，including specific implementation and gadgets search algorithm on ARM. Finally，a library sandbox technology is suggested to defense this kind of attack and secure the system.

[Keywords] return-oriented programming；ARM architecture；gadget search；library sandbox

0 引言

恶意篡改程序流攻击是当今主流安全威胁中最常见的攻击方式之一。所谓程序流控制攻击，是指攻击者利用软件漏洞往程序内存中写入超长数据，覆盖其返回地址，达到控制程序执行行为的目的。通常攻击者的 shellcode[1] 都会注入到用户的内存空间，在堆栈上执行恶意操作。为了防御此类攻击方式，操作系统中引入 W ⊕ X 保护机制，即数据执行保护 (DEP)，硬件上由 Intel 的 XD(Execute Disable bit) 技术和 AMD 的 ND(No-Execute Page-Protection) 技术支持。返回导向编程 (ROP) 正是基于这种防御技术提出的，通过总结 ROP 在 X86 平台上的攻击及防御技术，重点研究该攻击移植到 ARM 平台的具体技术，并提出了一种加载系统库沙盒技术来防御该攻击。

1 返回导向编程

返回导向编程的前身 return-into-libc 攻击利用溢出漏洞使程序跳转到系统函数中，组合系统调用来实现恶意操作。该攻击方式的缺陷只限利用现有函数而不能完成任意操作。在此基础上，Shacham 在 2007 年第一次提出了返回导向编程 [2]，这种技术通过搜索以 ret(0xc3) 或 retn(0xc2) 结尾的指令序列并组合成逻辑运算、函数调用、内存读取、条件执行等操作的 gadgets，再通过 gadgets 组合完成任意操作。近几年，Stephen Checkoway 等人 [3] 对 ROP 进行了深入研究，提出了变种 JOP 攻击 (Jump-oriented Programming，JOP)。该攻击回避使用 ret 指令构造 gadgets，而采用类 ret 指令序列，如 “ pop x；jmp *x ” (x 为 X86 寄存器)。该种攻击能够有效突破检测 ret 指令特征的防御机制，如 ret 指令频率、ret 影子栈等。为了防御该攻击，研究者提出了各种检测机制，表 1 分析比较了检

表 1　X86 平台 ROP 防御技术比较

名称	描述	ROP	JOP
ROPdefender	影子栈，检查 call-return 一致性，运用 pin 插桩工具	√	X
DROP	动态时检测 ret 指令，识别 gadget 并记录数量，达到一定阈值报警，运用 valgrind 插桩工具	√	X
Ret-less	编译器级别，在系统库中消除 ret 指令	√	X
G-Free	修改编译器级别，消除 unaligned 跳转，保护 aligned 跳转	√	√
DeROP	Gadgets 转换，把 ROP shellcode 转变成为 none-ROP shellcode	√	√
ROPscan	逐字节扫描输入，如存在疑似 gadget 地址，则模拟执行，检测其行为是否为 ROP 攻击	√	√
ASLR	系统库，堆栈地址随机化	√	√

收稿日期：2012-07-19
作者简介：钱逸，1987 年生，女，硕士研究生，研究方向：网络攻防；王轶骏，1980 年生，男，讲师，研究方向：网络攻击与防御；薛质，1971 年生，男，教授，博士生导师，研究方向：网络与信息安全。
* 基金项目：国家自然科学基金资助项目 (批准号：61171173)。

测 ROP/JOP 攻击方法的检测能力。利用 ROP/JOP 技术无需注入代码就能完成攻击，绕过了 DEP 机制，因此在溢出攻击中被广泛运用，MIDI 文件漏洞 CVE-2012-0003 的利用即用了 ROP 技术。

以上介绍了基于 X86 平台的 ROP 攻击和防御技术。经研究，这种攻击技术同样也适用于精简指令集架构，如 ARM、SPARC。下面介绍基于 ARM 平台的 ROP 攻击和防御技术。

2 ARM 平台的 ROP 攻击技术

2.1 ARM 架构简介

相较 X86 平台的 CISC 架构，ARM[4] 的 RISC 架构在指令集及实现方式上存在着很大的差异。不同于 X86 的不定长指令，ARM 指令是 32/16 位定长指令集，且 4/2 字节对齐。ARM 没有 ret 指令，但它提供对程序寄存器 PC 的直接操作，任何一条写 PC 寄存器指令都可以当作跳转指令来执行。如“ ldr pc，[r0] ”，表示把 r0 地址所存储的值赋给 PC，并跳转到 PC 所指向的目标地址继续执行。ARM 支持两种指令集，即 32 位 ARM 指令集和 16 位 Thumb 指令集。两种指令集根据跳转地址的最后一个 bit 切换，address<0>=1，跳转到 Thumb 指令集；address<0>=0，跳转到 ARM 指令集。

2.2 ARM 平台 ROP 攻击原理

Kornau 在文献 [5] 里首次提出了 ARM 架构 ROP 攻击，证明了搜索应用程序中可执行代码和链接库代码足够完成任意操作的图灵完整性。之后 Lucas 等人提出了用间接跳转指令 (blxreg) 指导完成的 JOP 攻击，并在 android 2.0 系统[6] 上呈现了该攻击。要完成一次 ROP 攻击，必须借助系统库或可执行区域内的指令片段的组合。这些 gadget 的最大特点是最后一条指令是一条跳转指令，如“ bllr ”，上下两个 gadgets 之间通过内存操作指令，如“ ldmfdsp!，{r4，r7，lr} ”，由上一个 gadget 写栈参数，下一个 gadget 读栈参数完成。通过精心设计栈上下一块 gadget 执行地址和函数参数，指导 gadgets 完成一次完整的攻击。运用 ARM 间接跳转指令 blx 的 ROP 攻击示例如图 1 所示。

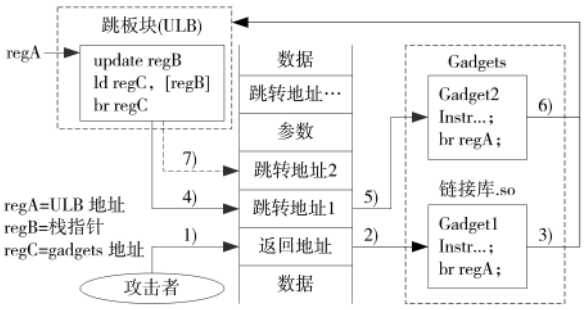

图 1 ARM 平台 ROP 攻击流程

整个 ROP 攻击流程如下：

1) 攻击者首先利用应用程序漏洞将 gadget1 的地址覆盖返回地址，并布置好 regA、regB，当函数返回时，程序流转到受攻击者控制的第一个 gadget1 执行，对应于图 1 中的 1) 和 2)。

2) gadget1 的最后一条指令 brregA 跳转到 ULB 块 (regA 指向 ULB 地址)。由于每块 gadget 最后一条指令是间接跳转，需要一个跳板块 (trampoline)ULB(update-load-branch) 来桥接两个 gadgets。ULB 块一般是 1-3 条指令的组合，完成更新栈寄存器、装载下一个 gadget 块地址，以及跳转的功能，对应于图 1 中的 3)。

3) ULB 首先更新 regB 使得 regB 指向地址 1，装载 load regC，[regB]，此时，regC 中存放了 gadget2 的地址，branch regC 跳转到 gadget2 执行，对应于图 1 中的 4) 和 5)。

4) 6) 和 7) 为 3) 和 4) 的重复。

上述流程依次下去直至 ROP 链执行完毕。

2.3 gadgets 搜索算法

目前为止，还没有一个公开的基于 ARM 指令集的 gadgets 搜索工具，Kornau 提出了一种平台无关的基于 REIL 语言的自动搜索 gadgets 算法，但整个算法很复杂。

借助于 Kornau 和 Schacham 的搜索方法，文中在 ARM 平台下进行了如下实现：首先在链接库文件中搜索 rret 指令特征流 (“ bl/blxreg ”[0xff\0x2f\0xe1]、“ pop {..pc} ”[0x80\0xbd\0xe8]，这些指令等同于 X86 上 ret 指令)，4 字节反向汇编 ARM 指令流，到达阈值或无前向指令停止。由于 ARM 指令的定长特性，每次只需搜索向前 4 bytes 即可，搜索算法如下：

可用指令搜索算法 (1)：

```
Search_rret_Sequence(code，depth) // 搜索 rret 指令
Creat a trie_seq
    For pos from 1 to code_len  // 搜索整个 code 文件
        If 3-bytes@pos in rret_seq  //rret_seq 见上二进制流
    Call Build_Valid (pos，trie_seq，depth)
```

可用指令搜索算法 (2)：

```
Build_Valid(pos，root，depth) // 反向汇编 depth 长度
    for i=1 to depth
        if dis_arm(pos-4) is valid
    add_seq(pos，trie_seq)
```

经过上述算法搜索得到可用序列的信息，进一步分析组合这些指令使其成为完成一个特定操作 (如 mov/add/and/…) 的 gadget，为以后 ROP 链 Exploit 漏洞打下基础。

根据上述算法，文中在 ARM 平台下做了仿真实验，实验环境搭建如下：

主机环境：intel(R) Core(TM) i5-2300 CPU @ 2.80 GHz，4.00 GB 内存。

虚拟机环境：VMware? Workstation 7.1.2

build-301548，ubuntu-10.10-desktop-i386，android-sdk_r18-linux 模拟器，android 2.3 模拟平台。

表 2 给出了在 android2.3 系统 library 库 ARM 的 rret 指令分布情况。

表 2 ARM 系统库 rret 分布总结

链接库文件	#bx 指令	#blx 指令	#pop{pc} 指令	Gadgets 数目
libandroid.so	17	1 239	317	1 587
libc.so	812	620	1 848	3 779
libdl.so	85	42	18	180
libm.so	180	27	200	577
libterm.so	0	196	43	316
libutils.so	6	949	1 015	2 537

3 ARM 平台的 ROP 防御技术

基于 X86 平台的 ROP 防御技术已研究得很纯熟，从编译器到运行时，从静态分析到动态分析，在各层面都提出了有效的防御检测方法。但由于平台的差异性，这些防御技术并不能直接运用在 ARM 架构上。ZhiJun Huang 等人[7]提出了一种根据 AAPCS 函数调用规则的运行时检测方法，但缺陷是并非所有的函数调用都按照 AAPCS 规则，且这种检测技术开销很大。HristoBojinov 等人提出了一种在 android 系统上地址随机化的机制，但只能作用在系统更新时。基于此，下面提出一种加载系统库沙盒技术来防御 ARM 平台的 ROP 攻击。

由于 ROP 攻击是将 gadgets 在内存中的绝对地址安排到堆栈，截获程序流时跳转到该地址去执行代码，如 addrA=0xafd13f10(android2.3 系统 libc.so 的加载基地址为 0xafd00000)。假设 addrA 的基地址变化了，那么地址 0xafd13f10 上的 gadget 已不是原来的攻击代码，从而使得整个 ROP 链失效，攻击失败。整个系统模块设计如图 2 所示。

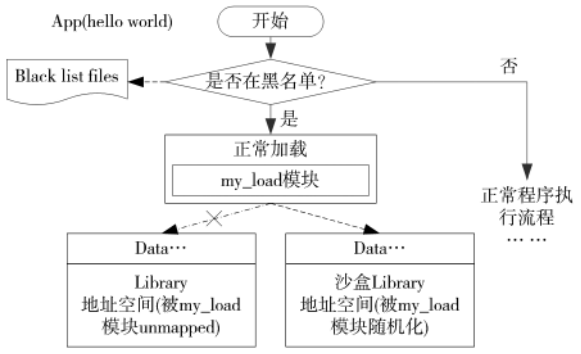

图 2 沙盒防御系统

假设系统模块的作用前提是有一份潜在的黑名单，在这份黑名单里面的程序较容易受到攻击。首先，当一个应用程序启动时，加载器判断该程序是否在黑名单里面，如果不在，则按正常的应用程序加载执行；如果在，则由依附在加载器上的 my_load 模块 unmap 掉原来的系统库地址，重新 mmap 一个新的地址空间 (secret space)，从而当攻击者访问原来的 addrA 地址时发生错误而导致攻击失败，如图 3 所示。My_load 模块的设计需要修改该程序相对应的 GOT 和 PLT 表，保证程序在没有受到攻击的时候能够正常运行。

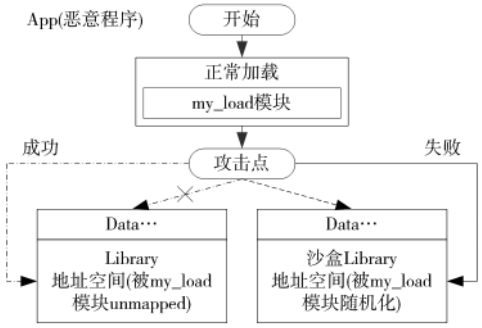

图 3 ROP 攻击比较

4 结语

地址随机化机制几乎让 ROP 攻击无用武之地，但攻防技术本就是一场“猫和老鼠”的游戏，继续深入研究 ROP 具有一定的现实意义，特别是在跨平台领域内，如 ARM、SPARC、PowerPC 等。由于平台之间的差异性，如何开发一套平台无关的自动化 ROP 链工具加速 Exploit 过程具有很大的现实意义，未来的研究方向将重点放在自动构建 ROP 链上。

参考文献

[1] 陈悦，薛质，王轶骏 . 针对 Shellcode 变形规避的 NIDS 检测技术 [J]. 信息安全与通信保密，2007(1)：99-103.

[2] SHACHAM H. The Geometry of Innocent Flesh on the Bone：Return-into-libc Without Function Calls(on the x86)[EB/OL]. (2007-11-20)[2012-4-3]. http：//dl.acm.org/citation.cfm?id=1315313.

[3] CHECKOWAY S，DAVI L，DMITRIENKO A，et al. Return-oriented Programming Without Returns[C]// Proceedings of the 17th ACM Conference on Computer and Communications Security. NY：New York，2010.

[4] 丛欣 . ARM 处理器在网络安全领域中的应用 [J]. 信息安全与通信保密，2011(5)：41-42.

[5] KORNAU T. Return Oriented Programming for the ARM Architecture. Masterthesis[D]. Germany：Ruhr-University Bochum，2009.

[6] 刘仙艳 . 移动终端开放平台—Android[J]. 通信技术，2011，44(4)：50-53.

[7] HUANG ZhiJun，ZHENG Tao，LIU Jia. A Dynamic Detective Method against ROP Attack on ARM Platform[C]//SEES 2012. Zurich，Switzerland：IEEE Publications，2012：51-57.