计算机科学
COMPUTER SCIENCE

面向 Cisco IOS 的 ROP 攻击检测方法

李鹏宇，刘胜利，尹小康，刘昊晖

引用本文

李鹏宇，刘胜利，尹小康，刘昊晖. 面向 Cisco IOS 的 ROP 攻击检测方法[J]. 计算机科学, 2022, 49(4): 369-375.

LI Peng-yu, LIU Sheng-li, YIN Xiao-kang, LIU Hao-hui. Detection Method of ROP Attack for Cisco IOS[J]. Computer Science, 2022, 49(4): 369-375.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210300153

面向 Cisco IOS 的 ROP 攻击检测方法

李鹏宇[1,2]　刘胜利[1,2]　尹小康[1,2]　刘昊晖[2]
1 数学工程与先进计算国家重点实验室　郑州 450000
2 战略支援部队信息工程大学　郑州 450000
(void_yu0359@163.com)

摘　要　Cisco IOS(Internetwork Operating System)作为 Cisco 路由器的专用操作系统,其由于硬件条件限制,在设计时更加注重性能而忽视了系统安全,导致无法有效检测面向返回地址编程(Return-Oriented Programming,ROP)的攻击。针对传统的 ROP 防护技术在解决 Cisco IOS 防护上存在的缺陷,提出了一种基于返回地址内存哈希验证的方法,能够对面向 Cisco IOS 的 ROP 攻击进行有效检测,并对 ROP 攻击代码进行捕获。通过分析现有针对 ROP 攻击的防护机制的优缺点,在紧凑型影子内存防护思想的基础上,将传统的影子内存存储模式改造为基于哈希的内存查找模式,增加了返回地址内存指针的记录作为哈希查找的索引,提高了影子内存查找效率,同时能够抵御由于内存泄露导致的影子内存篡改。在 Dynamips 虚拟化平台的基础上设计实现了 CROPDS 系统,对所提方法进行了有效验证。与现有方法对比,所提方法在通用性和性能上均有提升,并能够捕获到攻击执行的 shellcode。

关键词:Cisco IOS;ROP 攻击;影子栈;哈希表;攻击检测
中图法分类号　TP393

Detection Method of ROP Attack for Cisco IOS

LI Peng-yu[1,2],LIU Sheng-li[1,2],YIN Xiao-kang[1,2] and LIU Hao-hui[2]
1 State Key Laboratory of Mathematical Engineering and Advanced Computing,Zhengzhou 450000,China
2 Information Engineering University,Zhengzhou 450000,China

Abstract　Cisco IOS (Internet operating system) is a special operating system of Cisco router. Due to the limitation of hardware conditions,it pays more attention to the performance and ignores the system security in the design,which makes it unable to effectively detect the attack of return address oriented programming (ROP). Aiming at the defects of traditional ROP protection technology in Cisco IOS protection,a method based on return address memory hash verification is proposed,which can effectively detect the ROP attack on Cisco IOS and capture the attack code. By analyzing the advantages and disadvantages of the existing protection mechanisms against ROP attacks,on the basis of the idea of compact shadow memory protection,the traditional shadow memory storage mode is transformed into a hash based memory search mode,and the record of the return address memory pointer is added as the index of hash search,which improves the efficiency of shadow me-mory search and can resist shadow memory tampering caused by memory leakage. Based on the Dynamips virtualization platform,the CROPDS system is designed and implemented,and the method is verified effectively. Compared with the previous methods,it improves the generality and perfor-mance,and can capture the shellcode of attack execution.

Keywords　Cisco IOS,ROP attack,Shadow stack,Hash table,Attack detection

1　引言

在网络安全领域,缓冲区溢出一直是引发应用程序安全问题的重要原因。利用缓冲区溢出,攻击者通过构造特定数据覆盖相关内存,可以篡改程序的正常执行流程[1]。缓冲区溢出的攻击利用方式主要有两种:代码注入和代码重用。随着 W⊕X 等技术的出现,代码注入方法已经无效[2]。传统的代码复用技术主要利用面向 libc 库的编程(return-to-libc),将原本的函数返回地址覆盖为 libc 库中的函数地址(如 system 函数等),同时在栈上预设好相关参数从而调用库函数实现恶意行为。但攻击者可以使用的攻击序列只能是 libc 中的库函数,所以这种攻击的能力有限。随后发展出面向返回

到稿日期:2021-03-15　返修日期:2021-07-09
基金项目:国家重点研发计划(2019QY1300);科技委基础加强项目(2019-JCJQ-ZD-113)
This work was supported by the National Basic Research Program of China(2019QY1300) and Science & Technology Commission Foundation Strengthening Project(2019-JCJQ-ZD-113).
通信作者:刘胜利(dr_liushengli@163.com)

地址的编程技术，首先在程序代码中寻找带有特殊内存操作指令的短小代码片段 gadget，通过控制函数返回地址跳转将 gadget 串联为一段具有完整逻辑功能的程序块以达到攻击目的。ROP 攻击已经被证明是图灵完备的，逐步演变为缓冲区溢出攻击的主流方法。

思科作为目前市场上最大的网络设备制造商[3]，其产品安全对互联网安全的影响很大。Cisco IOS 作为思科路由器设备的操作系统，同样存在被缓冲区溢出的可能。从 2003 年 Black Hat 大会上 Linder 等[4]开始提出关于思科漏洞利用的若干问题及缓冲区溢出方法开始，陆续出现关于 Cisco IOS 安全机制和脆弱性的研究[5-7]，其中涉及到缓冲区溢出的漏洞利用方法大多使用了 ROP 技术。

由于硬件条件限制，Cisco 路由器设备在设计时更加注重性能而忽视了系统安全，Cisco IOS 本身并没有专门针对代码执行攻击的检测能力。为了解决 Cisco IOS 对缓冲区溢出漏洞特别是 ROP 攻击的检测能力不足的问题，本文分析了现有 ROP 攻击防护机制的优缺点，针对 Cisco IOS 的指令特点和运行机制，提出了一种基于栈返回地址内存哈希校验的 ROP 攻击运行时检测方法。该方法能够在不需要前置静态分析的情况下，对面向 Cisco IOS 的 ROP 攻击进行有效监测，对 ROP 攻击的过程代码具备捕获能力。

2 相关研究

2.1 ROP 防护与检测

2.1.1 ASLR 技术

针对面向返回地址的攻击，最有效的防御机制是地址空间布局随机化（Address Space Layout Randomization，ASLR）。ASLR 对进程的堆、栈、代码、共享库等地址在程序每次运行时进行随机化，能够极大地增加攻击者分析定位代码正确位置的难度。但是实际上 ASLR 并不是绝对的[8]，尤其是在实现上，各模块的内部结构依然相对固定，或者存在一定的关联关系[9]，攻击者可以通过未启用的 ASLR 部分或者不彻底的随机化部分实现同样的攻击效果[10]。

2.1.2 栈金丝雀

在函数调用时向栈内插入一个小整数（金丝雀值），在函数返回时检查这些值是否改变，以此判断程序执行流是否发生改变。其缺点是只考虑栈溢出顺序覆盖的假设，而不是真正检查返回地址的篡改情况，因此金丝雀不能防止函数返回地址前的缓冲区溢出，通过调试或者泄露金丝雀值的方法能够绕过该保护机制[11]。

2.1.3 程序控制流完整性

程序控制流完整性（Control-Flow Integrity，CFI）保护由加州大学和微软公司于 2005 年首次提出[12]，其核心思想为通过分析程序的控制流图，将程序的运行限制在合法的范围内。目前大多数控制流解决方案都是无状态的[13]，它们独立验证每个控制流的传播，而不区分控制流图中的不同路径。其中完全精确的静态 CFI[14]被认为是最严格的无状态策略，通过静态分析建立函数 F 的合法返回地址集合，在返回时允许被返回到任何调用 F 的函数地址处，这类策略容易遭受控制流弯曲攻击[12]。有状态的 CFI 思想为将函数的返回限制在活动的调用点上，通过分支记录等技术提取上下文敏感信息[15-17]，以缩小合法返回的范围。虽然现代 64 位英特尔处理器中有可用于辅助分支记录的硬件功能，但其仍然被证明是不适合采用的[18]。

2.1.4 影子栈技术

影子栈技术是被设计用来对抗 ROP 攻击的有效方法。其核心原理为维护一块相对独立的内存作为程序的栈空间副本，当函数返回时通过对比返回地址与影子栈中的地址的一致性来检测 ROP 攻击。影子栈通常分为两种类型：平行影子栈和紧凑型影子栈[19]平行影子栈利用直接映射技术完全备份程序栈空间内容，其优点是实现简单、维护方便、兼容性好，缺点是消耗的内存空间较大。由于位置相对固定，平行影子栈容易遭受攻击篡改。紧凑型影子栈利用间接映射方案，并不完全复制程序栈空间，而是仅记录程序的返回地址部分，其优点是减少了内存开销，缺点是需要维护影子栈与程序栈的一致性，程序出现 setjmp/longjmp 调用和 C++异常处理过程中的不规则的栈展开时无法做到完全匹配，在调整影子栈指针时会带来额外开销[13]。

2.2 Cisco IOS 相关

2.2.1 Cisco 防护绕过

Cisco IOS 是思科路由器专用操作系统，其中 IOS 镜像文件是由一个巨大的静态链接而成的二进制文件，通常直接运行在嵌入式硬件环境下，CPU 采用 PowerPC 或 MIPS 两种架构。Cisco IOS 同样实现了 DEP 和 ASLR 机制，但仍存在缺陷，其中 ASLR 并不彻底，在加载过程中仅仅改变了段基址，各个组件的相对位置变化很小且在一定的范围内，给 ROP 攻击提供了可能。同时 Cisco IOS 在系统加载的过程中首先会从 ROMMON[20]中获得引导，而 ROMMON 在内存中的位置相对固定且未进行随机化，因此攻击者可以使用 ROMMON 中的内存区域构造 ROP 攻击链绕过 ASLR，实现任意地址写功能，从而进行相关恶意执行。

鉴于 Cisco IOS 本身并没有专门针对代码执行篡改进行防护，研究者提出了一些防护措施和攻击检测方法。Chen 等[21]在 Dynamips[22]仿真器上实现了一种基于动态污点分析的 Cisco IOS 漏洞攻击检测方法。通过修改 Dynamips 的路由器仿真源码，在虚拟网卡接收网络报文时进行污点引入，并且对虚拟 CPU 执行的每条污点操作的相关指令执行进行污点跟踪，以发现接收的报文中的疑似恶意行为。此方法虽然能够记录网络数据在内存中的污点传播，但指令数据被污染并不一定代表恶意攻击，因此该方法误报率较高。Liu 等[23]利用固件静态分析方法构造合法转移地址集合，采用动静态结合的方法，通过判断 Cisco IOS 执行过程中的函数返回地址与影子栈是否一致以及是否符合合法转移地址集合来确定是否遭受攻击。此方法并没有考虑函数的异常控制转移和影子栈数据的安全性，并且依赖于对 IOS 固件的静态分析，通用性不佳。

2.2.2 Cisco IOS 函数调用与返回

Cisco IOS 采用 MIPS 或 PowerPC 两种 CPU 架构，均为精简指令集，其指令长度固定，具有相似特点。本文重点关注函数的调用及返回时的指令特点。其中 MIPS 和 PowerPC

架构中的子函数调用与返回时的返回地址出入栈操作指令如表 1 所列。

表 1 MIPS 和 PowerPC 架构子函数调用与返回时的栈操作指令

Table 1 Stack operation instructions of sub function call and return in MIPS and PowerPC architecture

架构	子函数调用前返回地址入栈指令	子函数返回前返回地址出栈指令	子函数返回跳转指令
MIPS	sw $ra,xx($sp)	lw $ra,xx($sp)	jr ra
PowerPC	mflr r0; stw r0,xx(r1)	lwz r0,xx(r1); mtlr r0	blr

以 MIPS 指令集为例,第 31 号寄存器 ra 为子函数的返回地址寄存器,永远存放着函数调用指令的返回地址;29 号寄存器 sp 为栈指针寄存器,在函数开始时通过 addiu $sp,-YY 指令将 sp 寄存器值减去 YY,向下开辟函数的栈空间。当父函数调用任何一个子函数时,父函数会将子函数的返回地址保存在 ra 寄存器中,此时子函数的最后一条指令为 jr ra(寄存器直接跳转到 ra)。在子函数调用孙子函数前,子函数会将寄存器 ra 中保存的子函数的返回地址保存在子函数的栈空间,指令为 sw $ra,XX($sp),同时寄存器 ra 的值变更为孙子函数的返回地址。当孙子函数调用结束后子函数返回前,子函数会从自己的栈空间取回相应的返回地址到 ra 寄存器,指令为 lw $ra,XX($sp),再利用 jr ra 进行返回地址跳转。

3 基于内存哈希验证的动态检测方法

本节介绍基于内存哈希验证的动态检测方法,主要包括 ROP 攻击的定位和 ROP 攻击代码的捕获。

3.1 基本定义

定义 1 在子函数调用前,执行返回地址入栈时,存储返回地址的栈内存指针为 *point*,实际入栈的返回地址值为 *ret*,即子函数合法返回地址。则在 MIPS 和 Power 两种平台架构下的 *point* 和 *ret* 分别由式(1)和式(2)计算得出:

$$point=\begin{cases} xx+\$sp, & \text{MIPS} \\ xx+r1, & \text{PowerPC} \end{cases} \tag{1}$$

$$ret=\begin{cases} ra, & MIPS \\ r0, & \text{PowerPC} \end{cases} \tag{2}$$

其中,xx 为内存偏移值,sp 和 $r1$ 分别为 MIPS 平台和 PowerPC 平台中栈寄存器地址寄存器的值,ra 和 $r0$ 分别为 MIPS 平台和 PowerPC 平台中返回地址寄存器的值,后文含义相同。

定义 2 在子函数返回前,返回地址出栈时,存储返回地址的栈内存指针为 $point'$,实际出栈的返回地址值为 re''。则在 MIPS 和 Power 两种平台架构下的 $point'$ 和 ret' 分别由式(3)和式(4)计算得出:

$$point'=\begin{cases} xx'+\$sp', & \text{MIPS} \\ xx'+r1', & \text{PowerPC} \end{cases} \tag{3}$$

$$ret'=\begin{cases} (xx'+\$sp'), & \text{MIPS} \\ (xx'+r1'), & \text{PowerPC} \end{cases} \tag{4}$$

定义 3 子函数返回跳转的目的返回地址 *jret*。则在 MIPS 和 Power 两种平台架构下的 *jret* 由式(5)计算得出:

$$juet=\begin{cases} ra, & \text{MIPS} \\ lr, & \text{PowerPC} \end{cases} \tag{5}$$

3.2 ROP 攻击定位方法

ROP 攻击定位的目标是要找到出现函数非法返回时的程序运行地址,并记录当前函数的非法返回地址。其思路为当程序运行到函数调用时构造合法返回地址副本,当程序运行到需要返回时进行返回地址的验证,如果出现不一致,则此时的程序运行地址和函数返回地址即为所求。

其中合法返回地址副本包含当前的栈内存指针 point 和实际入栈的返回地址值所组成的键值对。

3.2.1 合法返回地址记录

设计哈希表作为函数返回地址副本对 *point* 和 *ret* 进行记录,采用链地址法避免 hash 冲突。其中:

$$index=\text{hash}_1(point) \tag{6}$$

$$value=\text{hash}_2(ret) \tag{7}$$

index 为哈希表中数据的存放位置索引,这里通过 hash_1 函数对 *point* 进行哈希计算得出;*value* 为哈希表中 *index* 位置处的值,通过对正常函数返回地址 *ret* 做哈希 hash_2 后得出。

3.2.2 攻击判定

函数合法返回的判断依据为保证式(8)为真,即同时满足 *point* 与 $point'$ 相等且 *ret* 与 ret' 相等,否则为遭受 ROP 攻击。

$$(point==point')\&\&(ret==ret') \tag{8}$$

由于本文采用哈希表对合法返回信息进行存储,因此函数合法返回的判断依据为保证式(9)为真,即经过 hash 加密后的 *point* 与 $\text{hash}_1(point')$ 相等且加密后的 *ret* 与 $\text{hash}_2(ret')$ 相等,否则为遭受 ROP 攻击。

$$(index==\text{hash}_1(point'))\&\&(value==\text{hash}_2(ret')) \tag{9}$$

3.2.3 检测流程

ROP 攻击定位流程如算法 1 所示,具体流程图如图 1 所示。

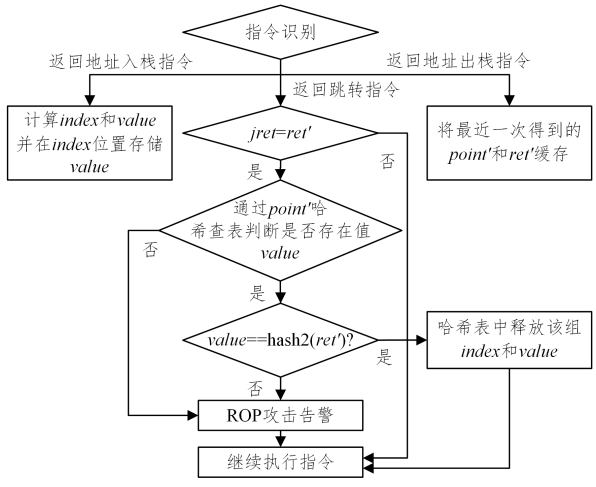


图 1 ROP 攻击定位算法流程

Fig. 1 Process of ROP attack location algorithm

算法 1 ROP 攻击定位

步骤 1 进行指令识别。如果为返回地址入栈指令则跳转到步骤 2；如果为返回地址出栈指令则跳转到步骤 3；如果为返回地址跳转指令则跳转到步骤 4；否则不做处理继续执行。

步骤 2 根据式(6)和式(7)计算当前 index 和 value 的值并在哈希表中的 index 位置存储 value。

步骤 3 缓存 point′和 ret′值保持这两个值始终由最近一次的返回地址出栈指令得出。

步骤 4 判断 jret 与 ret′是否相等。如果相等，说明当前函数返回跳转的 jret 值来自栈空间保存的返回地址，即需要验证的返回地址，转至步骤 5；如果不相等，说明 jret 跳转为叶子函数的返回，不需要验证，转至步骤 8。

步骤 5 尝试在哈希表中 $hash_1$(point′)位置处取出 value 并判断 value 是否存在。如果存在，进行下一步判断，转至步骤 6；如果不存在，说明遭受 ROP 攻击，转至步骤 8。

步骤 6 比较 value 值是否与 $hash_2$(ret′)相等。如果相等，说明此次为正常返回转至步骤 7；否则说明返回地址发生篡改，转至步骤 8。

步骤 7 删除 hash 表中该次返回地址的副本，继续执行指令。

步骤 8 继续执行指令。

3.3 攻击代码捕获方法

攻击者为了实现一定的功能或者植入代码的执行，会进行控制流的劫持。对于基于栈溢出的 ROP 攻击，即通过溢出篡改保存在栈上的返回地址，在函数返回时，会跳转到修改后的目标地址，执行构造好的 gadget 攻击链代码实现特定的功能。为了避免程序崩溃，完整的 ROP 攻击在 gadgets 链利用完毕时需将寄存器恢复，使栈空间平衡，同时将程序执行流转移到首次篡改返回地址之前的执行流中，以免程序发生异常。图 2 为一次常见的 ROP 攻击示例，程序在函数 FuncB 处发生 ROP 攻击，控制流被转移至 gadget 链，gadget 执行完之后，攻击者将控制流交回到函数 FuncB 的上层调用函数 FuncA 中。

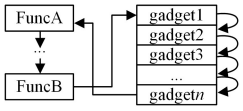


图 2 ROP 攻击流程图

Fig. 2 ROP attack process

基于上述思想，从发生返回地址篡改到控制流交回正常执行序列的过程中的程序执行即为完整的攻击代码。设计 ROP 攻击代码捕获算法，算法的流程如图 3 和算法 2 所示。

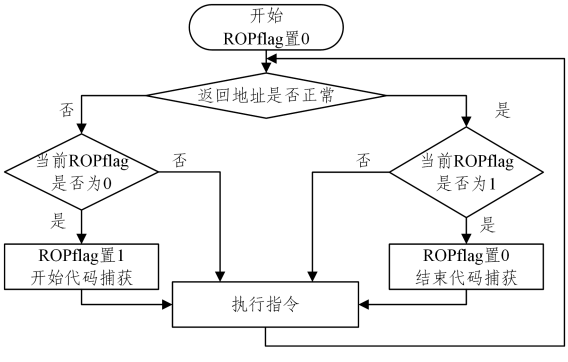


图 3 攻击代码捕获流程

Fig. 3 Attack code capture process

图 3 中，ROPflag 为当前是否遭受 ROP 攻击的标志，0 表示未遭受攻击，1 表示处于 ROP 攻击过程中。

算法 2 ROP 攻击代码捕获算法

步骤 1 初始时 ROPflag 置 0。

步骤 2 判断当前返回地址是否正常，若正常则转到步骤 3，否则转到步骤 4。

步骤 3 判断 ROPflag 是否为 1。如果是，说明当前正处于 ROP 攻击中，此时为停止攻击信号，将 ROPflag 置 0；如果否，说明当前未处于 ROP 攻击中，则不做处理继续执行指令。

步骤 4 判断 ROPflag 是否为 0。如果是，说明当前未处于 ROP 攻击中，此时为 ROP 开始攻击信号，将 ROPflag 置 1；如果否，说明当前正处于 ROP 攻击当中，攻击记录已经开始，则不做处理继续执行指令。

4 实验分析

4.1 实验环境

为了验证本文方法的有效性，在 Dynamips 平台的基础上设计实现了 CROPDS 系统，检测系统架构如图 4 所示。系统完成了对 Cisco 路由器的虚拟化仿真，能够加载执行 Cisco IOS，生成虚拟路由器。在对 Cisco IOS 虚拟化执行过程中对指令执行进行监控，其中检测告警模块应用本文 ROP 攻击定位算法实现了对 ROP 攻击的定位告警，代码捕获模块运用本文 ROP 攻击代码捕获方法实现了对 ROP 攻击过程的记录。

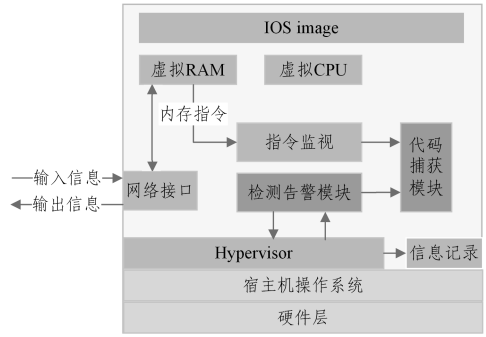


图 4 检测系统架构

Fig. 4 Detection system architecture

如图 5 所示设置测试环境。其中服务器 B 运行 CROPDS 系统，路由器 R1 为 CROPDS 虚拟出的路由器。

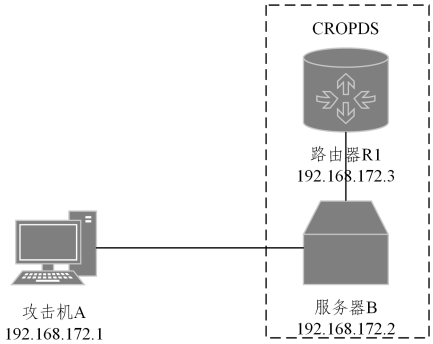


图 5 测试环境

Fig. 5 Test environment

通过端口映射技术，R1 可以借助 B 的网口与外界通信。主机 A 为攻击验证主机，与虚拟路由器 R1 处于同一网络中，

利用攻击机 A 对 R1 发起 ROP 攻击，同时观察 CROPDS 系统是否能够准确发现该次攻击，并捕获相关攻击代码。本实验使用的所有攻击测试代码均为真实有效的 EXP，且攻击方式均为 ROP 攻击。

4.2 可行性验证

下面以 CVE-2017-6736[24] 漏洞为例对本文方法进行说明。该漏洞是一款 Cisco IOS 的 SNMP 协议实现漏洞，属于缓冲区溢出漏洞，广泛存在于 IOS12.0-12.4，15.0-15.4 版本之中。攻击者通过向路由器发送精心构造的 SNMP 数据包来触发缓冲区溢出漏洞并执行 shellcode。本次实验选择存在上述漏洞的 IOS 版本为“c1700-entbase-mz.124-5”的 c1700 路由器进行 ROP 攻击验证。图 6 为事先利用 ida 解析“c1700-entbase-mz.124-5”IOS 固件在相关位置的攻击代码片段。图 7 为系统对该次 ROP 攻击的检测及代码捕获的告警信息。

```
.text:8049BADC loc_8049BADC:                          # CODE XREF: sub_8049B770+E4↑j
.text:8049BADC                                        # sub_8049B770+150↑j
.text:8049BADC                lwz       r0, 52(r1)
.text:8049BAE0                mtlr      r0
.text:8049BAE4                lmw       r27, 28(r1)
.text:8049BAE8                addi      r1, r1, 48
.text:8049BAEC                blr
.text:8049BAEC # End of function sub_8049B770
```

(a)首次异常返回代码

```
.text:800ADFB4 loc_800ADFB4:                          # CODE XREF: sub_800ADEE0+AC↑j
.text:800ADFB4                mr        r3, r27
.text:800ADFB8
.text:800ADFB8 loc_800ADFB8:                          # CODE XREF: sub_800ADEE0+40↑j
.text:800ADFB8                lwz       r0, 36(r1)
.text:800ADFBC                mtlr      r0
.text:800ADFC0                lmw       r27, 12(r1)
.text:800ADFC4                addi      r1, r1, 32
.text:800ADFC8                blr
.text:800ADFC8 # End of function sub_800ADEE0
```

(b)gadget1 的代码

```
.text:80127E50                stw       r3, 0(r27)
.text:80127E54                lwz       r0, 36(r1)
.text:80127E58                mtlr      r0
.text:80127E5C                lmw       r27, 12(r1)
.text:80127E60                addi      r1, r1, 32
.text:80127E64                blr
.text:80127E64 # End of function sub_80127E1C
```

(c)gadget2 代码

```
.text:810B4F30                lwz       r0, 516(r1)
.text:810B4F34                mtlr      r0
.text:810B4F38                lmw       r24, 480(r1)
.text:810B4F3C                addi      r1, r1, 512
.text:810B4F40                blr
.text:810B4F40 # End of function sub_810B49B8
```

(d)gadget3 代码

图 6 ida 解析 c1700-entbase-mz.124-5 固件的相关位置代码片段

Fig. 6 Ida parsing code fragment of c1700-entbase-mz.124-5 firmware related location

从告警信息上可以看出，系统发现程序运行到 0x8049baec 时首次出现返回地址不正确，给出告警。此时的异常指令为 lwz r0，r1，52。出现异常的栈内存地址为 0x82bff63c，原本此处存储的函数返回地址 0x80497860 被篡改为 0x800adfb4。与图 6(a)所示 ida 固件解析中的 gadget0 相关代码吻合，证明定位准确。

图 7 后 17 条代码为攻击捕获代码，说明发生 ROP 攻击后程序又运行了 17 条指令。在恶意函数返回后程序控制流被转移到 0x800adfb4 执行，之后又在 0x80127e54 和 0x810b4f30 处分别做了两次栈上的返回地址篡改。达到攻击效果后在 0x810b4f40 处将控制流转移到正常的函数调用序列中，至此攻击结束。攻击代码与图 6(b)、图 6(c)、图 6(d)吻合，说明攻击代码捕获正确。

```
Jan 07 12:36:01.106 cpu->cpu_thread:2970498816 router:R1 rop message:
current ia:8049baec,current_insn:blr
danger ia:8049badc, danger insn:80010034 lwz r0, r1, 52, danger point:
82bff63c,dangerretaddr:800adfb4,original retaddr:80497860
Jan 07 12:36:01.106 cpu->cpu_thread:2970498816 router:R1
ROP Execution process:
ia:800adfb4 insn:7f63db78 or r3,r27,r27
ia:800adfb8 insn:80010024 lwz r0,r1,36
ia:800adfbc insn:7c0803a6 mtlr r0
ia:800adfc0 insn:bb61000c lmw r27,r1,12
ia:800adfc4 insn:38210020 addi r1,r1,32
ia:800adfc8 insn:4e800020 bclr 14h,0h
ia:80127e50 insn:907b0000 stw r3,r27,0
ia:80127e54 insn:80010024 lwz r0,r1,36
ia:80127e58 insn:7c0803a6 mtlr r0
ia:80127e5cinsn:bb61000c lmw r27,r1,12
ia:80127e60 insn:38210020 addi r1,r1,32
ia:80127e64 insn:4e800020 bclr 14h,0h
ia:810b4f30 insn:80010204 lwz r0,r1,516
ia:810b4f34 insn:7c0803a6 mtlr r0
ia:810b4f38 insn:bb0101e0 lmw r24,r1,480
ia:810b4f3c insn:38210200 addi r1,r1,512
ia:810b4f40 insn:4e800020 bclr 14h,0h
```

图 7 c1700 路由器 ROP 攻击告警信息

Fig. 7 Warning information of ROP attack on c1700 router

4.3 通用性与性能测试

为了测试 CROPDS 的通用性和性能，实验选择 CtaintDetect[21]，CIDS[23] 和 Dynamips 原型系统作对比实验。其中 Dynamips 为 Cisco 路由器仿真原型系统，Dynamips 本身不具备攻击检测能力，可以作为运行时效的参考基准。前两者和本系统均为基于 Dynamips 原型系统进行开发，均具备对 Cisco-IOS 的漏洞相关攻击检测的能力，因此可作为通用性参考。

为了验证 CROPDS 的通用性，实验选择了 FTP，IP，SNMP 3 款 Cisco IOS 协议实现上缓冲区溢出漏洞，针对不同 IOS 版本路由器进行 ROP 攻击测试。其中每款漏洞需在不同平台路由器下分别选择 10 种不同 IOS 版本进行攻击，记录检测结果。攻击检测结果如表 2 所列。可以看到，对 3 款漏洞共 50 款 IOS 的 ROP 攻击检测中，本文方法的检测成功率达到了 96%，优于前两款系统，且具备前两者所不具备的攻击代码捕获能力。该实验说明了本文方法具备对 IOS 版本类别和漏洞种类的 ROP 攻击检测的通用性。

为了测试 CROPDS 的运行性能，在漏洞攻击测试中分别对 CtaintDetect，CIDS，CROPDS 和 Dynamips 系统的响应时间 t_i 做记录。以 Dynamips 虚拟化原型系统的运行响应时间 t 为基准单位 1，采用时间消耗比作为参照指标，各平台系统的时间消耗比计算方式为 t_i/t。

时间消耗比的测试结果如图 8 所示，可以看出 CtaintDetect 的时间消耗约为 Dynamips 的 4 倍，CIDS 的时间消耗约为 Dynamips 的 1.5 倍，本文方法 CROPDS 的时间消耗约为 Dynamips 的 1.3 倍，处理性能优于以上两者。其中 CtaintDetect 由于污点分析算法需要从异常数据包的引入开始反复

跟踪内存污染传递情况，内存读写消耗大，因此耗时较高。CIDS需要依赖提前建立IOS的静态分析库作为合法返回参考，而由于查表范围为整个IOS的合法返回集合，因此耗时较长。

表2 ROP攻击实验结果的统计

Table 2 Test result statistics of ROP attack

漏洞版本	协议	路由器型号	IOS版本范围	IOS数量/个	CtaintDetect		CIDS		CROPDS		CROPDS攻击代码捕获成功次数	CROPDS攻击代码捕获成功率/%
					ROP检测成功次数	ROP检测成功率/%	ROP检测成功次数	ROP检测成功率/%	ROP检测成功次数	ROP检测成功率/%		
CVE-2007-2586	FTP	2600	11.3-12.4	10	7	70	10	100	10	100	10	100
CVE-2007-0480	IP	2600	9.x,10.x,11.x 12.x	10	8	80	9	90	9	90	9	100
CVE-2017-6736	SNMP	1700	12.0-12.4,15.0-15.6	10	7	70	9	90	10	100	9	90
		2600	12.0-12.4,15.0-15.6	10	8	80	7	70	9	90	8	88.8
		3745	12.0-12.4,15.0-15.6	10	7	70	8	80	10	100	10	100
汇总				50	37	74	43	86	48	96	46	95.8

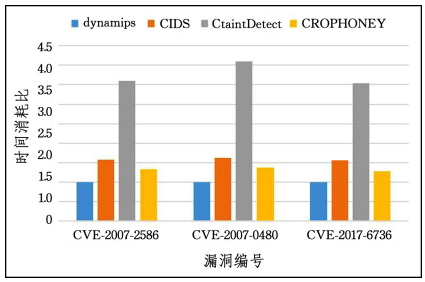


图8 时间消耗比的测试结果

Fig. 8 Time consumption comparison test results

5 分析与讨论

本节从安全性、查找速度、通用性等方面对比前人方法，并讨论了本文代码捕获的思想。

(1)返回地址副本的安全性

与影子栈类似，本文方法也是将合法的返回地址副本作为ROP攻击检测依据。不同的是，在影子栈的实现中，如果副本内存在泄露，即攻击者能够通过同时篡改返回地址和验证副本的值来实现对影子栈的绕过，本文方法会对函数合法返回地址进行hash，即副本中存储的是hash加密后的函数返回地址，使得攻击者无法通过分析内存中的函数合法返回地址来定位副本中合法返回地址的位置 *index*，也就无法在合法的存储位置上篡改 *ret*，从而保证了返回地址副本的安全可靠。

(2)查找速度

上文已分析了当紧凑影子栈遇到程序异常时，为了能够验证地址的合法性，通常的做法是遍历影子栈直到找到与之匹配的返回地址为止，这会带来极大开销。本文方法采用hash计算直接定位内存的方法进行查表，在没有碰撞的情况下只需一次查表，较大地提高了查表性能。但本文方法在函数调用深度增加或频繁存在异常中断时，函数的返回可能不及时，需要缓存的地址增多，哈希散列表占用较大，出现哈希碰撞的可能性增加，极端情况下哈希散列表被占满，查表速度等同于遍历影子栈。因此在后续的优化中需要考虑通过调整哈希函数和存储空间来解决哈希碰撞问题。

(3)通用性

本文方法为动态检测，不需要对IOS进行前置静态分析，不需要提前构建合法的函数返回地址集合，不存在静态分析构建合法集合不准确的问题，不存在ASLR开启后静态分析前置条件信息缺失导致无法检测的问题。相对于紧凑型影子栈无法有效应对setjmp/longjmp等异常处理，本文采用内存地址和返回地址双认证，解决了因中断异常等造成的影子栈错序问题，不需要实时维护副本与程序栈帧的对应，因此对异常处理也具有适用性。

(4)代码捕获能力

本文所构建的攻击代码捕获思想建立在攻击者对一次ROP攻击进行完整的利用的前提下，完整的ROP攻击利用为了不造成被攻击对象的程序崩溃，通过平衡栈空间保证不破坏程序后续执行，需将程序的执行逻辑跳转到之前的执行流中。本文方法会将此过程的攻击代码全部抓取，其中包含了ROP跳转逻辑和shellcode执行逻辑，也包含了shellcode执行结束到恢复正常执行流的逻辑。目前本文只是将此过程的汇编指令进行了记录，其中寄存器的值并没有一并输出，gadget连接的程序片段阅读起来依然复杂，下一步可以考虑将每步指令中寄存器的值一并输出以提高分析者对攻击代码的分析效率。

结束语 本文根据Cisco IOS指令特点，提出了一种基于栈返回地址内存哈希校验的ROP攻击检测方法，设计哈希表对函数返回地址进行记录和验证，增加了返回地址内存指针的记录作为哈希查找的索引，提高了影子内存查找效率，并且能够防止返回地址副本的篡改。对本文方法在虚拟仿真器上进行了实现，通过真实ROP攻击样例对本文方法进行了验证。实验证明利用本文提出的方法，可以在不依赖静态分析的前提下，对ROP攻击触发点准确定位，并具备一定的ROP攻击代码的捕获能力。运用该方法实现的系统CROPDS，通过模拟执行Cisco IOS生成的虚拟路由器，可作为蜜罐直接在互联网中进行部署，对Cisco IOS所遭受的ROP攻击执行流程进行捕获，便于研究者对缓冲区溢出漏洞的发现和漏洞利用。下一步将从两方面进行改进，一是根据Cisco IOS在

实际执行过程中的函数调用情况，探索合适的哈希函数和哈希存储空间大小，在保证哈希存储空间合理的情况下尽可能地提高查找速度；二是在代码捕获信息输出上适当增加动态执行信息，以便于攻击代码的阅读。

参 考 文 献

[1] CHAUM D. Untraceable electronic mail, return addresses, and digital pseudonyms[J]. Communications of the ACM, 1981, 24(2):84-90.

[2] SZEKERES L, PAYER M, WEI T, et al. SoK: Eternal war in memory[C]// Proceedings of the 34th IEEE Symposium on Security and Privacy. IEEE, 2013:48-62.

[3] IDC. Global Ethernet Switch and Router Markets Deliver Mixed Results in Q2 2020, According to IDC[EB/OL]. (2020-09-03)[2021-01-24]. https://www.idc.com/getdoc.jsp?containerId=prUS46830820.

[4] LINDER F. Design and software vulnerability in embedded system[EB/OL]. (2003-04-25)[2021-01-12]. https://www.blackhat.com/presentations/bh-usa-03/bh-us-03-FX.pdf.

[5] LYNN M. The holy grail: Cisco IOS shellcode and exploitation techniques[EB/OL]. (2005-07-29)[2021-3-14]. https://mirror.die.net/banned/lynn-cisco.pdf.

[6] MUNIZ S. Killing the myth of Cisco IOS rootkits: DIK(Da IOS rootkit)[EB/OL]. (2008-06-25)[2021-03-19]. http://www.orkspace.net/secdocs/Conferences/EuSecWest/2008/Cisco IOS Rootkits-paper.pdf.

[7] LINDER F. Cisco IOS router exploitation[EB/OL]. (2009-06-22)[2021-01-02]. https://www.blackhat.com/presentations/bh-usa-09/LINDNER/BHUSA09-Lindner-RouterExploit-PA-PER.pdf.

[8] HUANG N, HUANG S G, PAN Z L, et al. Automatic analysis to vulnerability of ASLR[J]. Journal of National University of Defense Technology, 2020, 42(2):162-170, 185.

[9] EVTYUSHKIN D, PONOMAREV D, ABU-GHAZALEH A. Jump over ASLR: attacking branch predictors to bypass ASLR[C]// Proceedings of 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). 2016:1-13.

[10] PAYER M, GROSS T R. String oriented programming: when ASLR is not enough[C]// Proceedings of the 2nd ACM SIGPLAN Program Protection and Reverse Engineering Workshop. ACM, 2013:1-9.

[11] ALPS233. Canary Stack protection mechanism[EB/OL]. (2019-10-25)[2021-01-12]. https://blog.csdn.net/ALPS233/article/details/102736299.

[12] ABADI M, BUDIU M, ERLINGSSON Ú, et al. Control-flow integrity[C]// Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05). ACM, 2005:340-353.

[13] LILJESTRAND H, NYMAN T, GUNN L J, et al. PACStack: an Authenticated Call Stack[C]// Proceedings of the 30th USENIX Security Symposium. 2020.

[14] CARLINI N, BARRESI A, PAYER M, et al. Control-flow bending: On the effectiveness of control-flow integrity[C]// Proceedings of the 24th USENIX Security Symposium (USENIX Security '15). USENIX, 2015:161-176.

[15] REN D, QIAN C, SONG L, et al. Effificient protection of path-sensitive control security[C]// Proceedings of the 26th USENIX Security Symposium (USENIX Security '17). USENIX, 2017:131-148.

[16] HU H, QIAN C X, CARTER Y, et al. Enforcing unique code target property for control-flow integrity[C]// Proceedings of the 25th ACM Conference on Computer and Communications Security (CCS 2018). ACM CCS, 2018:1470-1486.

[17] VICTOR V, DENNIS A, ENES G, et al. Practical Context-Sensitive CFI[C]// Proceedings of the 22nd ACM Conference on Computer and Communications Security. ACM CCS, 2015:927-940.

[18] MARTÍN A, MIHAI B, ULFAR E, et al. Control-flow integrity principles, implementations, and applications[J]. ACM Trans., 2009, 13(1):4:1-4:40.

[19] BUROW N, ZHANG X P, PAYER M. SoK: Shining light on shadow stacks[C]// Proceedings of the 40th IEEE Symposium onSecurity and Privacy. IEEE, 2019:985-999.

[20] WANG J Z, CAI R J, LIU S L. Research on the Protection Mechanism of Cisco IOS Exploit[C]// Proceedings of 4th International Conference on Data Mining, Communications and Information Technology (DMCIT 2020). Asia Pacific Institute of Science and Engineering: Chengdu Sherlock Education Consulting Co., Ltd., 2020, 1584(1):012045.

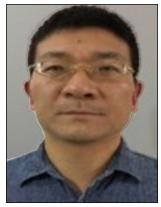
[21] CHEN L G, LIU S L, GAO X, et al. A Vulnerability Attack Detection Method Based on Dynamic Taint Analysis for Cisco IOS [J]. Journal of Chinese Computer Systems, 2014, 35(8):1798-1802.

[22] ANUZELLI G, FILES N, EMULATION P, etal. Dynamips/Dynagen: tutorial[EB/OL]. (2011-10-07)[2021-01-13]. http://materias.fi.uba.ar/7543/2010-02/download/DynamipsTutorial.doc.

[23] LIU S L, ZOU R, PENG F, et al. A Method for Detecting Cisco IOS Flow Monitoring[J]. Journal of Xi'an Jiaotong University, 2015, 49(12):65-70, 111.

[24] DHS, CISA. CVE-2017-6736[EB/OL]. (2017-03-09)[2021-01-02]. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-6736.

LI Peng-yu, born in 1993, postgraduate. His main research interests include network device security and network attack detection.



LIU Sheng-li, born in 1973, Ph.D professor. His main research interests include network device security and network attack detection.

（责任编辑：柯颖）