基于动态分析的控制流劫持攻击检测

吴小王[1]，方　勇[1]，贾　鹏[1]，刘露平[2]，王　炎[1]
(1. 四川大学网络空间安全学院，成都 610065；2. 四川大学电子信息学院，成都 610065)

摘　要：控制流完整性策略能够有效地防御控制流劫持类攻击，但复杂的控制流图构建使该策略的实际部署非常困难. 为了有效地针对控制流劫持攻击的检测手段，提出一种基于程序异常行为来检测控制流劫持攻击的方法，并基于该方法实现了控制流劫持攻击检测系统. 实验表明，该方法能够有效地检测出由控制流劫持攻击造成的各种程序异常行为，基于该方法的攻击检测系统运行效果良好，能够方便的部署到实际应用环境中.

关键词：控制流劫持；攻击检测；动态分析；程序异常；控制流完整性
中图分类号：TP309　　**文献标识码**：A　　DOI：10.19907/j.0490-6756.2021.032004

Control flow hijacking attack detection based on dynamic analysis

WU Xiao-Wang[1], *FANG Yong*[1], *JIA Peng*[1], *LIU Lu-Ping*[2], *WANG Yan*[1]
(1. School of Cyber Science and Engineering, Sichuan University, Chengdu 610065, China;
2. College of Electronics and Information Engineering, Sichuan University, Chengdu 610065, China)

Abstract: The control flow integrity (CFI) strategy can effectively defend against control flow hijacking attacks, but the complex control flow graph construction makes the actual deployment of this strategy very difficult. In order to solve the problem, a method is proposed to detect control flow hijacking attacks by identifying abnormal program behaviors, and a control flow hijacking attack detection system is implemented based on this method. The experiment results show that this method can effectively detect various abnormal program behaviors caused by control flow hijacking attacks; the attack detection system works well and can be easily deployed in the actual application environment.

Keywords: Control flow hijacking; Attack detection; Dynamic analysis; Program exception; CFI

1　引　言

网络空间是一个没有硝烟的战场，从 1988 年的 Morris 蠕虫病毒到 2017 年的 WannaCry 勒索病毒足以可见网络空间安全的严峻态势. 据 CNVD[1]最新的收录漏洞类型统计，应用程序漏洞占比高达 58.45%，并且漏洞数量在逐年增多. 越来越多的高级持续性威胁(Advanced Persistent Threat，APT)利用应用程序漏洞执行破坏行为[2]，对网络空间安全造成巨大的威胁. 尽管各大软件厂商在不断改进和完善软件开发质量管理，但软件漏洞问题仍无法彻底消除.

应用程序漏洞的利用方式多种多样，而控制流劫持攻击是最常见的一种. 控制流劫持攻击允许攻击者破坏程序的控制数据，通常将执行流重定向到攻击者自己的注入代码. 通过执行恶意代码能够完全控制程序和系统，造成极大的危害. 目前的操作系统中部署了地址空间布局随机化(Address

收稿日期：2020-07-24
基金项目：CCF-启明星辰"鸿雁"科研资助计划(CCF-VenustechRP2017002)；国家重点研发计划(2017YFB0802900)
作者简介：吴小王(1997－)，男，四川达州人，硕士研究生，研究方向为二进制安全. E-mail：1493553260@qq.com
通讯作者：贾鹏. E-mail：pengjia@scu.edu.cn

Space Layout Randomization,ASLR)、数据执行保护(Data Execution Protection,DEP)、结构化异常处理安全校验(Safe Structured Exception Handling,SafeSEH)等安全机制,一定程度上缓解了控制流劫持攻击对软件和系统的危害. 但是,因为系统环境的多样性,攻击者仍然能够找到绕过这些安全机制的方法. 因此,如何快速高效地发现攻击威胁并及时采取应对措施一直是学术界的研究热点.

由 Abadi 等人于 2005 年提出的控制流完整性 CFI(Control Flow Integrity)是目前防御控制流劫持攻击的主要技术[3]. CFI 的核心思想是获取程序间接转移指令的合法目标并以此构造应用程序的控制流图 CFG(Control Flow Graph),强制程序在 CFG 的范围内运行,能有效地防御控制流劫持攻击. 自提出 CFI 后,对控制流劫持攻击防御的研究有很大的推进作用,出现了如 binCFI[4]、CCFI[5]为代表的粗粒度的 CFI;为提高 CFI 防御能力的上下文敏感的 CFI[6],基于数据约束的 vCFI[7],操作系统内核层面的 KCoFI[8],专门针对 ROP 攻击的 ROPecker[9],随机选择特定跳转进行验证的 RCFI[10].

动态污点分析[11]是检测漏洞攻击的另一种主要技术,该技术跟踪程序数据在程序内部的传播,分析程序数据是否处于敏感位置,如常见的函数返回地址,从而检测控制流劫持攻击.

CFI 策略虽然能够有效的防御控制流劫持攻击,但是难以在实际环境中部署. 其主要原因有两点:(1) 是 CFG 的构造困难,完整的 CFG 构造需要依赖程序源码,对于复杂的程序而言更是难以实现. 虽然没有源码情况下通过逆向分析等技术也可以构建 CFG,但是对于采用了加壳[12]、混淆[13]等保护技术的二进制程序,构建 CFG 难以实现;(2) 是需要检查程序中的每一个间接控制转移,细粒度的检测会引起非常大的开销. 此外,动态污点分析技术在实际检测中存在过污染、欠污染等问题,所以效率较低且精度不高.

针对上述问题,本文提出了一种新的检测控制流劫持攻击的方法:在分析控制流劫持攻击原理的基础上,总结了控制流劫持攻击的行为特征,然后基于二进制插桩技术,通过动态分析应用程序的异常行为,来检测控制流劫持攻击. 与现有的动态污点分析技术和基于 CFI 策略的控制流劫持攻击检测方法相比,本文方法解决了二进制文件分析需要依赖源码[14]和修改二进制文件的问题,因此具有良好的移植性;另外,所有的检测分析仅在程序运行时进行,不需要静态分析和预处理的过程,具备良好的性能开销,可以方便地部署于各种系统环境中.

2　控制流劫持攻击研究

2.1　控制流劫持攻击原理

文献[15]总结了控制流劫持攻击的基本流程(如图 1). 一个完整的控制流劫持攻击包含以下 5 个步骤:构造特定的攻击载体;触发并利用应用程序中的内存破坏漏洞;劫持程序的控制流;绕过 DEP、ASLR 等安全防护;最后控制程序执行流转向恶意代码,实现漏洞攻击.

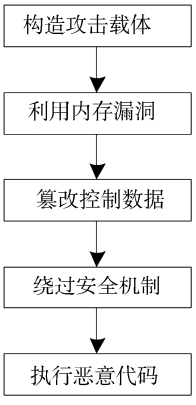

图 1　控制流劫持攻击流程图

Fig. 1　Flow chart of control flow hijacking attack

2.2　控制流劫持攻击行为

2.2.1　*执行恶意代码*　执行恶意代码主要有代码注入和代码复用两种方式. 注入的恶意代码被称为 shellcode,一般布置到应用程序的栈空间或者堆空间中. 栈空间用于存放函数参数的值、局部变量的值和函数调用保存的栈帧信息,正常程序不会从栈空间解析指令执行,常见的程序漏洞如缓冲区溢出[16],则会将恶意代码布置到栈空间中执行. 堆是程序运行过程中动态使用的内存空间,用于存放数据,也用于存放动态生成的指令,因此应用程序存在堆空间执行指令的情况.

攻击者除了利用自己构造的恶意代码,还有可能利用程序本身的代码,这种利用方式称为代码重用技术[17],如常见的 ROP[18]攻击. ROP 攻击利用程序本身的指令片段,这些指令片段称为 gadget,每个 gadget 由 ret 指令结尾. 通过 ret 指令将分散的指令片段连接起来,能实现与 shellcode 相同的功能.

ASLR 技术增加了 ROP 构造的难度，DEP 技术使得注入代码无法执行. 因此，代码注入通常与代码复用相结合，首先使用 ROP 关闭 shellcode 所在空间的数据执行保护，然后执行 shellcode.

2.2.2　造成程序异常　实际的漏洞利用过程中构造的攻击载体一般是针对特定的操作系统和目标程序，而现实中的系统环境多种多样，因此有不少的漏洞攻击会因为环境不同致使应用程序出现内存访问异常甚至崩溃的情况.

漏洞攻击除了被动触发程序异常，还有主动触发异常并利用异常的情况. 当应用程序出现异常，Windows 系统下会调用结构化异常处理函数 SEH (Structured Exception Handling,)进行异常处理. 图 2 展示了 SEH 链表结构.

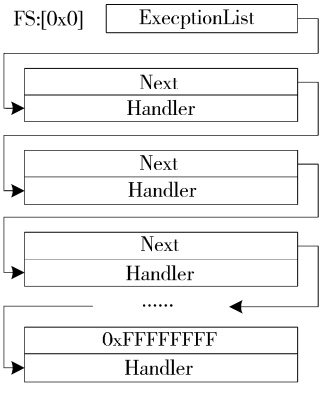

图 2　SEH 链表
Fig. 2　SEH chain

SEH 链表的每一个节点都包含 Next 和 Handler 两个成员，其中，Handler 指向具体的异常处理函数，Next 则指向 SEH 链表的下一个节点，最后一个节点的 Next 指向为 0xFFFFFFFF. 攻击者可以通过触发异常，然后覆盖 SEH 链表中的 Handler 地址来劫持程序的控制流，间接地破坏了 SEH 链表的完整性.

3　控制流劫持攻击检测方法

本文通过对控制流劫持攻击原理的研究，分析了在不同的漏洞利用攻击情况下程序的异常行为. 提出了基于动态分析的控制流劫持攻击检测方法，该方法通过动态获取程序运行时信息，针对不同的程序异常行为分别制定检测方法.

3.1　栈执行检测

程序的内存空间可以分为模块地址空间(程序加载的库文件和可执行程序本身)、栈空间和堆空间等 3 个区域. 栈空间主要存放的是函数调用时相关的信息，并不会存放代码指令，所以若发现函数执行流程跳转到栈中，则说明存在漏洞攻击行为. 如果不位于栈空间，则可能是位于堆空间中的恶意代码执行. 由于堆空间可能执行应用程序动态生成的代码，因此不能直接判定为恶意代码执行. 但是，通常在执行堆空间恶意代码前会使用 ROP 关闭 shellcode 所在内存空间的数据执行保护，因此，可以通过检测 ROP 防御部分堆空间恶意代码执行. 图 3 为栈执行检测流程图.

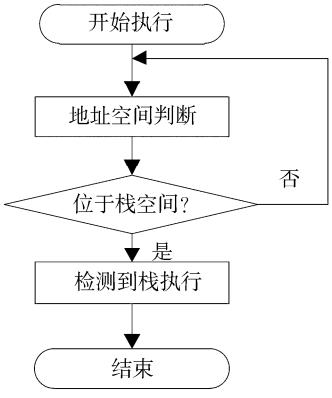

图 3　栈执行检测
Fig. 3　Stack space code execution detection

程序执行过程中，获取当前线程的栈空间范围，如果执行的指令地址位于栈空间范围内，则检测到栈执行.

3.2　ROP 攻击检测

文献[18]表明程序本身的一些指令片段可以构造出图灵完备的代码. ROP 攻击可以利用程序本身的代码片段执行一些函数的功能，比如调用 VirtualProtect 函数改变内存空间的可执行权限，使得堆空间中的恶意代码能够执行，或者仅仅使用 ROP 就完成攻击行为. 图 4 为 ROP 攻击检测流程图.

ROP 攻击利用的指令片段称为 gadget，这些 gadget 一般由几条指令组成，并以 ret 指令结尾，ret 指令让这些 gadget 联系在一起，通过顺序执行这些 gadget 完成一系列功能操作.

程序正常执行过程中函数调用与函数返回一般是匹配的，虽然存在特殊的 ret 指令与 call 指令不匹配的现象，但不会出现连续的 ret 与 call 不匹配的现象. 文献[18]表明实际的 ROP 攻击至少有 17 个连续的 gadget，而程序正常执行最多可能存在 10 个连续的 gadget. 因此，为了减少误报率和漏报率，节省时间开销，将连续出现的 11 个 ret 异常识别为 ROP 攻击，具体检测流程如图 5，通过构造影子栈，检测到 11 个连续的异常 ret 指令，判定

为 ROP 攻击.

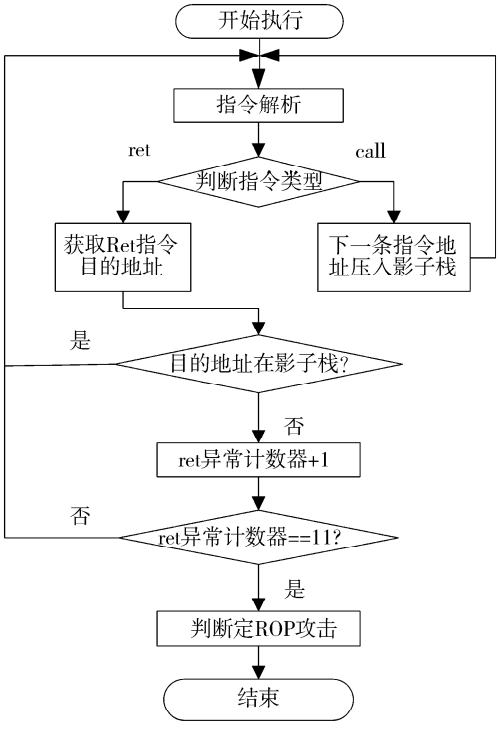

图 4　ROP 攻击检测
Fig. 4　ROP attack detection

3.3　SEH 异常检测

KiUserExceptionDispatcher 函数是 Windows 系统下异常处理的起点,该函数有两个参数 PEXCEPTION_RECORD 和 PCONTEXT,分别记录程序异常的具体信息和程序发生异常时的上下文信息. 控制流劫持攻击中程序异常分为以下两种情况:(1) 是攻击失败造成的异常;(2) 是攻击者为了利用 SEH 造成的异常. 对于第二种异常情况,程序在进行异常处理前 SEH 链表中的 handler 已经被恶意数据覆盖,导致 SEH 链表不再具备图 2 所示的完整结构. 因此,需要在 KiUserExceptionDispatcher 函数被调用时获得程序异常的具体信息和检查 SEH 链表的完整性. SEH 异常检测流程如图 5.

当捕获到 KiUserExceptionDispatcher 函数调用时,首先判断异常类型. 如果异常类型是"访问无效地址",直接判定为 SEH 异常; 如果不是,则接着检查 SEH 链表的完整性,如果 SEH 链表不完整则判定为 SEH 异常,否则,继续应用程序的异常监控.

在程序动态运行过程中,通过以上 3 种方法同时对程序的异常行为进行监控,结合程序运行时信息,实现对控制流劫持攻击的检测以及控制流劫持攻击地址定位和类型判断.

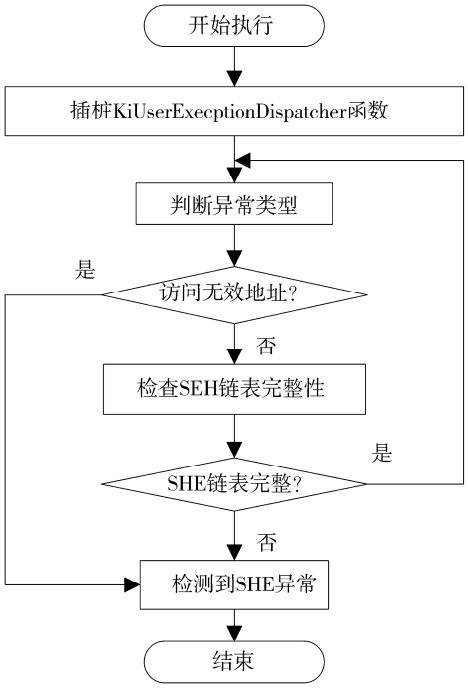

图 5　SEH 异常检测
Fig. 5　SEH detection

4　控制流劫持攻击检测系统设计与实现

Pin[19]是 Intel 公司开发的用于动态分析的二进制程序插桩工具,通过及时编译技术在程序原有代码中插入分析代码,从而获取程序动态运行时信息. 相较于基于 QEMU 虚拟环境的动态分析方式,基于 Pin 的动态分析技术能够直接在应用环境中使用. 因此基于 Pin 工具设计了控制流劫持攻击检测系统 CFHADS(Control Flow Hijacking Attack Detection System). 图 6 为 CFHADS 系统架构图.

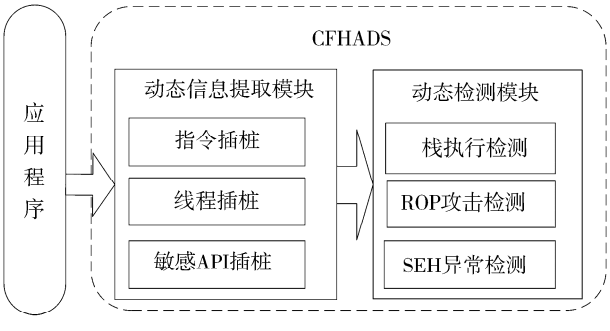

图 6　CFHADS 系统架构
Fig. 6　System structure of CFHADS

CFHADS 主要由动态信息提取模块和动态检

测 2 个模块组成. 其中，动态信息提取模块获取应用程序运行时信息以供动态检测模块实时分析. 使用 Pin 工具提供的 INS_AddInstrumentFunction 函数进行指令插桩、PIN_AddThreadStartFunction 函数进行线程插桩、RTN_InsertCall 函数进行敏感 API 插桩. 指令插桩主要分析 call 指令和 ret 指令，并结合线程插桩，获取应用程序线程创建的相关信息为每个线程构建影子栈；敏感 API 插桩主要分析程序异常处理过程中用到的函数并获取其参数. 此外，使用 Pin 工具提供丰富的 API，可为动态检测模块提供应用程序运行过程中任意时刻的上下文信息，如寄存器状态、内存数据等.

动态检测模块根据动态信息提取模块提供的程序运行时信息，实现本文提出的控制流劫持攻击检测方法中的栈执行检测方法、ROP 攻击检测方法和 SEH 异常检测方法. 我们通过对 3 种程序异常行为的检测实现对控制流劫持类攻击的检测，检测到攻击的同时，保存漏洞攻击的细节，用于后续分析.

5　实验与分析

我们将用真实的漏洞攻击样本对系统的功能和性能进行测试，表 1 为具体的实验环境配置.

表 1　实验环境配置

Tab. 1　Experimental environment configuration

名称	详细信息
处理器	Intel(R) Xeon(R) E3-1230 @3.40GHz
内存	8 GB
磁盘	1 000 GB
操作系统	Windows7 sp1 64-bit
虚拟环境	VMware Pro 12.5.9
插桩工具	Pin-2.14-71313-msvc10-windows

5.1　功能测试

为了验证 CFHADS 系统的攻击检测效果，选取了 26 个真实漏洞进行功能测试，这些漏洞都是针对的常用应用程序，包括 Microsoft office word、Microsoft office Excel、Foxit Reader、Adobe Acrobat Reader、IE 等. 漏洞利用样本主要来源于 exploit-db 和 metasploit. 具体检测结果如表 2 所示.

CVE-2012-0158 是针对常用软件 Microsoft office word 2007 的经典漏洞，该漏洞利用未开启 ASLR 保护的 msxml5.dll 中跳板指令绕过 DEP 的安全防护，使用了经典的跳板指令 jmp esp 控制应用程序在栈空间中执行恶意代码. 该漏洞实际的利用方式是 ret-to-libc，CFHADS 虽然没有检查 ret-to-libc 中最开始的异常跳转，但是准确检测到控制流劫持攻击的后续异常行为. 表 2 的检测结果显示，EDB-ID-42918 和 CVE-2018-7886 采用了与 CVE-2012-0158 相似的利用方式.

CVE-2010-1797 是针对 PDF 阅读器 Foxit Reader 的漏洞攻击，该漏洞采取了 SEH 利用的方式，CFHADS 的检测结果为内存访问异常＋SEH 利用＋栈执行，完整地记录了漏洞攻击从触发内存漏洞、劫持程序控制流到最后执行恶意代码的异常行为. 由于实际的系统环境与漏洞攻击的目标环境存在差异，因此检测结果中以内存访问异常居多. CVE-2010-2883 采用的攻击方式为 Heap Spray＋ROP，但是漏洞利用失败，应用程序 icucnv36.dll 偏移 0x2a715 处访问无效内存地址 0x0 导致崩溃退出. 虽然漏洞攻击没有完整实施后续的 ROP 攻击，CFHADS 仍准确地检测到漏洞攻击造成的程序异常.

实验结果表明，CFHADS 能检测到控制流劫持攻击各个环节的程序异常行为，虽然主要是检测栈执行、ROP 攻击和 SEH 利用，但是也间接防御了 ret-to-libc、堆利用等攻击，除了检测到完整的控制流劫持攻击，也能检测到漏洞利用失败的控制流劫持攻击.

5.2　性能测试

通过 Windows 系统下常用的 5 个应用程序测试 CFHADS 对应用程序的性能影响，图 7 展示了应用程序在没有附加 Pin、附加原生 Pin、附加 CFHADS 情况下的应用程序内存占用情况.

从图 7 可以看出，Pin 的运行开销约为原程序的 3 倍，而 CFHADS 的运行开销约为 Pin 的 1.2 倍. 因此，相对于基于虚拟机实现的漏洞攻击检测系统，CFHADS 可以直接部署到实际应用环境中进行实时的控制流劫持攻击检测.

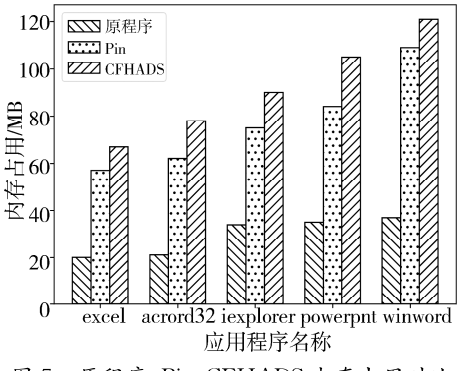

图 7　原程序、Pin、CFHADS 内存占用对比

Fig. 7　Memory usage comparison of original program, Pin and CFHADS

表 2　真实漏洞攻击检测结果

Tab. 2　Real sample detection result

漏洞编号	异常偏移	异常指令	异常类型
EDB-ID-40018	vuplayer. exe+0x539da	ret	rop、栈执行
EDB-ID-42918	qtgui4. dll+0x16677a	jmp esp	栈执行
EDB-ID-44382	faleemi. exe+0x1425	ret	c5:0x7a424f61
EDB-ID-44423	goldware. exe+0x2d0011	ret	c5、seh、栈执行
EDB-ID-44455	libdsm. dll+0xf5f8	mov edx,dword ptr [ecx]	c5:0x43434343
EDB-ID-44516	r. dll+0x175c1	ret	c5:0x41414141
EDB-ID-44828	zip-n-go49old. exe+0x4c70	ret	c5、seh、栈执行
EDB-ID-44903	audiograbber. exe+0x11d9d0	mov ecx,dword ptr [edx-8]	c5:0x 3f3f3f37
EDB-ID-45142	weddingslideshowstudio. exe+0x373c9	call esi	handler:0x41414141
EDB-ID-45176	ismartviewpro. exe+0x141e	ret	c5:0x42424242
CVE-2008-0015	msvidctl. dll+0x2d5a8	cmp dword ptr [ebx+8],esi	handler:0x0c0c0c0c
CVE-2009-0075	mshtml. dll+0x699cb	call dword ptr [ecx+0x4]	c5:0x0
CVE-2009-2655	mshtml. dll+0xc1b97	mov edx,dword ptr [edi]	c5:0x0
CVE-2009-4324	multimedia. api+0x42f8b	call dword ptr [edx+0x4]	栈执行
CVE-2010-0187	flash. ocx+0x2b064	mov dl,byte ptr [esi+eax]	c5:0x15555000
CVE-2010-0188	acroform. api+0x4b5a5d	ret	rop
CVE-2010-0806	jscript. dll+0x1b746	call dword ptr [ecx+0x8]	c5:0x40235093
CVE-2010-1797	foxit reader. exe+0x251150	ret	c5、seh、栈执行
CVE-2010-2883	icucnv36. dll+0x2a715	ret	c5:0x0
CVE-2011-0611	authplay. dll+0x1439d9	call dword ptr [edx+0x8]	c5:0x64004020
CVE-2012-0158	mscomctl. ocx+0x3c30	jmp esp	栈执行
CVE-2015-1641	wwlib. dll+0x9d30	mov esi,dword ptr ds:[ecx]	c5:0x7c38bd50
CVE-2015-2510	ogl. dll+0x8f14	movzx edx,byte ptr ds:[eax+0x1]	c5:0x7b19617a
CVE-2015-2521	excel. exe+0x51221	mov eax,dword ptr ds:[edx]	c5:0xffffffff
CVE-2018-7886	qt5core. dll+0x18a7b	jmp esp	栈执行
CVE-2010-3333	msvcr80. dll+0x14d3a	rep movs dword ptr es:[edi],dword ptr ds:[esi]	handler:0x78812890

6　结　论

本文提出了一种基于动态分析的控制流劫持攻击检测方法,在不依赖静态分析或者预处理的情况下检测应用程序运行时的控制流劫持攻击. 通过对控制流劫持攻击整个流程的详细分析,总结控制流劫持攻击在实际环境中造成的各种异常及行为特征,识别应用程序中的异常行为并判定为攻击. 本文基于动态分析技术实现了控制流劫持攻击检测系统,通过真实的漏洞攻击检测实验,表明该系统能够准确检测到控制流劫持攻击,良好的运行开销使得该系统能够部署到实际的应用环境中.

考虑到简单的 ret-to-libc 难以达到攻击目的、JOP 在实际漏洞利用中实现困难以及直接将控制流转移到堆空间中执行恶意代码实现复杂等情况,本文提出的方法没有针对以上攻击方式的具体检测手段,因此在遇见这些情况时可能会产生漏报. 另外,由于 Linux 系统与 Windows 系统的一些运行机制及处理机制的不同,本文提出的方法目前只适用于 Windows 系统. 后续研究计划中将进一步优化检测方法,提高控制流劫持攻击检测的全面性.

参考文献:

[1]　国家计算机网络应急技术处理协调中心. 国家信息安全漏洞共享平台[EB/OL]. [2020-05-24]. http://www. cnvd. org. cn/.

[2]　董刚, 余伟, 玄光哲. 高级持续性威胁中攻击特征的分析与检测 [J]. 吉林大学学报: 理学版, 2019, 57: 339.

[3]　Abadi M, Budiu M, Ligatti J. Control-flow integri-

ty[C]// ACM Conference on Computer and Communications Security. ACM，2005：340.

[4] Zhang M，Sekar R. Control flow integrity for COTS binaries[C]// Proceedings of the Usenix Conference on Security. [S. l.]：USENIX Association，2013.

[5] Mashtizadeh A J，Bittau A，Boneh D，*et al*. CCFI：Cryptographically enforced control flow integrity [J]. arXiv preprint arXiv，2014，13：941.

[6] Khandaker M R，Liu W Q，Naser A，*et al*. Origin-sensitive control flow integrity[C]//Proceedings of the 28th USENIX Security Symposium. [S. l.]：USENIX Association，2019.

[7] Jung D，Kim M，Jang J，*et al*. Value-based constraint control flow integrity [J]. IEEE Access，2020，8：50531.

[8] Criswell J，Dautenhahn N，Adve V. KCoFI：complete control-flow integrity for commodity operating system kernels[C]// Security and Privacy. [S. l.]：IEEE，2014.

[9] Cheng Y，Zhou Z，Yu M，*et al*. ROPecker：A generic and practical approach for defending against ROP attacks [C]//Proceedings of the 21st Network and Distributed System Security Symposium. [S. l.]：Internet Society，2014.

[10] Park M C，Lee D H. Random CFI (RCFI)：Efficient fine-grained control-flow integrity through random verification [J]. IEEE T Comput，2020，99：1.

[11] 陆开奎. 基于动态污点分析的漏洞攻击检测技术研究与实现[D]. 成都：电子科技大学，2013.

[12] 肖锦琦，王俊峰. 基于模糊哈希特征表示的恶意软件聚类方法 [J]. 四川大学学报：自然科学版，2018，55：469.

[13] 郭文，王俊峰. Windows 恶意代码动态通用脱壳方法研究 [J]. 四川大学学报：自然科学版，2018，55：283.

[14] 尹茗，张功萱. 基于源码分析的缓冲区溢出漏洞检测方法 [J]. 江苏大学学报：自然科学版，2016，37：450.

[15] 张超. 针对控制流劫持攻击的软件安全防护技术研究[D]. 北京：北京大学，2013.

[16] 邵思豪，高庆，马森，等. 缓冲区溢出漏洞分析技术研究进展[J]. 软件学报，2018，29：1179.

[17] 彭国军，梁玉，张焕国，等. 软件二进制代码重用技术综述[J]. 软件学报，2017，28：2026.

[18] Roemer R，Buchanan E，Shacham H，*et al*. Return-oriented programming：systems，languages，and applications [J]. ACM T Inform Syst Se，2012，15：1.

[19] Luk C K，Cohn R，Muth R，*et al*. Pin：building customized program analysis tools with dynamic instrumentation[J]. ACM Sigplan Notices，2005，40：190.

引用本文格式：

中　文：吴小王，方勇，贾鹏，等. 基于动态分析的控制流劫持攻击检测[J]. 四川大学学报：自然科学版，2021，58：032004.

英　文：Wu X W，Fang Y，Jia P，*et al*. Control flow hijacking attack detection based on dynamic analysis [J]. J Sichuan Univ：Nat Sci Ed，2021，58：032004.