基于动态插桩的缓冲区溢出漏洞检测技术研究

刘露平， 方 勇， 刘 亮， 龙 刚
（四川大学 电子信息学院，四川 成都 610065）

[摘 要]缓冲区溢出漏洞是一类常见的软件漏洞，其对计算机系统造成的危害非常大。本文针对这类漏洞提出一种基于二进制文件动态插桩并根据程序运行状态来判定缓冲区溢出的检测方法，并实现了基于该方法的检测系统。通过分析缓冲区溢出的原理以及常见攻击方法的特点，提出了基于覆盖返回地址、虚函数表、异常处理链表以及溢出后执行特定 API 的缓冲区溢出检测方法。实验表明该系统能有效检测到缓冲区溢出并定位溢出点从而辅助对漏洞原理进行分析。

[关键词]缓冲区溢出；动态检测；二进制插桩；程序状态
[中图分类号]TP393 [文献标志码]A [文章编号]1009-8054(2015)04-0080-03

Buffer Overflow Vulnerability Detection Technology based on Dynamic Instrumentation

LIU Lu-ping, FANG Yong, LIU Liang, LONG Gang
(College of Electronics and Information Engineering, Sichuan University, Chengdu Sichuan 610065, China)

[**Abstract**] Buffer overflow vulnerability, as a kind of common software vulnerability, would usually result in huge damage to computer system. In light of this, a method based on dynamic binary instrumentation is to detect buffer overflow vulnerability in accordance with the running states of program, and a detection system based on this method is also implemented. Based on the analysis of buffer overflow principle and characteristics of common attack-methods, a detection method based on the return address, virtual function table, exception handling linked-list and the carrying-out of specific API after overflow buffer is presented. Experiments indicate that this system can effectively detect buffer overflow and locate overflow point, so as to assist the analysis of vulnerability principle.

[**Key words**] buffer overflow; dynamic detection; binary-level instrumentation; program status

0 引言

缓冲区溢出漏洞是一类常见的软件安全性漏洞，广泛存在于各种操作系统、应用软件中，利用缓冲区漏洞进行攻击给网络安全带来了极大的威胁，由于其具有非常大的危害性，许多研究人员对此进行了深入的研究，并提出了各种针对缓冲区溢出的检测方法。

现有针对缓冲区溢出的检测和预警中，主要有以下几种方法：

1）基于编译器的运行时检查。如微软在其编译器 VS7.0 及以后版本中增加了 GS 编译选项，通过在堆栈中加一个 cookie，当函数返回时检测该 cookie 值是否被改变从而判断是否发生溢出。gcc 编译器的 StackGuard 插件则通过在栈上放探针来实现检测[1]。

2）系统/硬件级别的检测保护。在 Windows XP SP2 后，微软开始提供了 DEP（Data Execution Prevention）数据执行保护技术[2]，通过将堆和栈所属内存空间设置为不可执行属性，将数据和代码分开，从而有效阻止恶意代码的执行。

3）动态污点传播检测技术。通过将软件的外部输入数据标记为污点数据，然后利用动态插桩技术追踪记录污点数据在程序中的传播和扩展流程，结合符号执行等技术对程序引用污点数据的代码进行分析来判定是否发生缓冲区溢出[3-4]。

由于上述方法已被许多攻击者研究透彻，针对各种检测都有很多相应的绕过方法，在实际运用中并不能很好的阻止缓冲区溢出攻击。因此，本文通过对缓冲区溢出漏洞原理及其利用过程的分析，利用动态二进制插桩工具 Pin 来记录程序执行的整个过程，并结合缓冲区溢出特征来完成动态检测。实验表明该方法能有效检测和阻止缓冲区溢出漏洞攻击。

1 动态插桩检测原理

1.1 缓冲区溢出漏洞简介

缓冲区溢出漏洞是指当程序拷贝数据到特定的缓冲区时，由于预留空间较小多余的数据溢出后覆盖其他内存单元的值，导致内存破坏，程序发生崩溃。如果数据被经过特定构造，则通过溢出能劫持程序流程从而取得系统控制权。

缓冲区在系统中的表现形式多样，常见的缓冲区溢出包括栈溢出和堆溢出两种。在缓冲区漏洞利用中，一般会通过数据溢出

收稿日期：2014-11-27

万方数据

后覆盖函数返回地址，SEH(Structured Exception Handing)链表、以及函数虚表的方式来达到劫持程序流程的目的。

1.2 二进制动态插桩平台 PIN

Pin 是 Intel 公司开发的一个动态二进制探测框架，支持 Android、Linux、OSX 以及 Windows 等平台。具有易用、高效、可移植以及健壮性等特点，通过插桩引擎，将用户插入的分析代码和目标程序重新编译成新的目标文件后在虚拟机缓冲区中执行，从而可以动态获取程序运行的中间信息[5]。图 1 是 Pin 的整个框架：

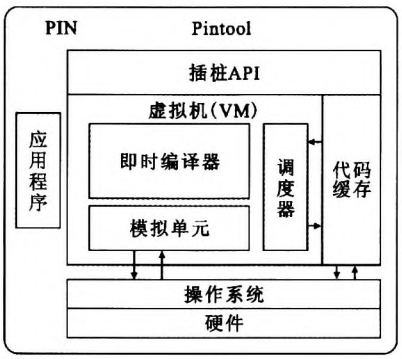

图 1 PIN 结构架构图

用户编写的分析工具相当于 Pin 的一个插件，统称为 Pintool。Pin 提供了指令、函数和映像三种插桩模式。在进行缓冲区溢出检测时，通过策略选择三种不同粒度的插桩方式，从而当指令执行时记录的指令类型、操作数、相关寄存器等信息；函数调用时记录下函数的参数、跳转地址、返回值信息；而当模块加载时获取到模块名称、以及库函数等信息。此外通过 pin 提供的事件进行插桩功能，可以在程序开始时插桩获取到进程的堆空间分布信息，在线程开始时插桩记录下线程的初始化堆栈空间分布信息。

通过插桩获取到程序运行状态信息后，结合缓冲区溢出的特征利用预先设定的规则来进行分析判断从而对缓冲区溢出漏洞攻击进行判定和预警[6]。

1.3 溢出检测的实现

缓冲区溢出后一般会采取覆盖函数返回地址、覆盖 SEH 链表或者虚函数表的方式进行利用。整个缓冲区溢出检测步骤如下：

(1)异常事件和特定行为捕获时机

Windows 系统利用 SEH 机制处理异常时，在用户态空间会调用 ntdll.dll 中的 DispathcException 这个函数进行异常的分发处理，可以对这个函数进行插桩来对程序异常进行捕获。另外溢出发生后执行 shellcode 时一般会调用 kernel32.dll 中的 LoadLibrary 和 GetProcAddress 这两个函数来获取系统 API 的地址，因此对这两个函数插桩可以捕获到 shellcode 调用这两个系统 API 时的动作。

(2)溢出检测流程

程序在进行调用时，通过 Call 指令将返回地址压入堆栈，而程序结束时通过 ret 指令将返回地址出栈后赋值给 EIP 继续执行，一旦发生溢出，则函数返回地址将被改写。因此方案中通过对 CALL、RET 指令插桩，并建立一个虚拟硬件堆栈，在执行 Call 指令进行函数调用时将返回地址压入虚拟硬件堆栈，RET 指令函数返回时从虚拟硬件堆栈中出栈返回地址并与真实堆栈的返回地址进行比较，判定是否相等来进行检测判断。

SEH 链表中的每个节点都是一个异常注册结构体(EXCEPTION_REGISTRATION)，其成员 NextS.E.H Recorder 指向下一个结构体，而 Exception handler 指向异常处理函数[7]。而基于 SEH 的缓冲区溢出攻击中，通常使溢出数据覆盖异常注册结构体后触发异常执行 shellcode。SEH 链表在溢出发生后被覆盖成如图 2 所示的形式：

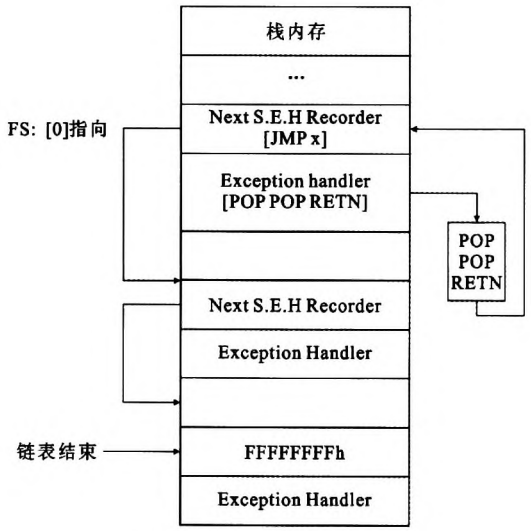

图 2 覆盖异常处理链表示意图

溢出发生后一般会将顶层异常注册结构体覆盖，因此 SEH 链表的完整性将得到破坏。一个完整的 SEH 链表满足以下特性：①整个链表节点全部位于堆栈中。②每个节点的异常处理函数指针不是位于堆栈中。③最后一个节点的指针为 0xffffffff。因此可以在异常事件发生时对 SEH 链表的完整性进行检测来判断是否发生溢出。

在多数堆溢出攻击中，在漏洞触发前会利用 Heap Spay(堆喷射)或其他方式进行内存空间的布局，通过在堆空间重复布置包含大量 NOP(空指令)和 shellcode 的数据库，溢出发生后将虚函数表覆盖成攻击者事先布置好的堆栈内存空间中[8]。因此可以对函数调用的 Call 指令进行插桩，在桩代码中获取其跳转目的地址检测其是为堆栈空间来进行判断。

此外在溢出攻击中，获取控制权后一般会调用系统 API 来完成功能，通用代码一般会调用 LoadLibraryA 和 GetProcAddress 等函数获取其他 API 函数地址，正常情况下这个两个函数的返回地

万方数据

址位于系统空间,而一旦在溢出发生后其返回地址位于堆栈等内存空间。因此对这两个函数进行插桩,判断其返回地址是否为堆栈空间来进行检测。根据以上分析,整个检测流程如下图3所示:

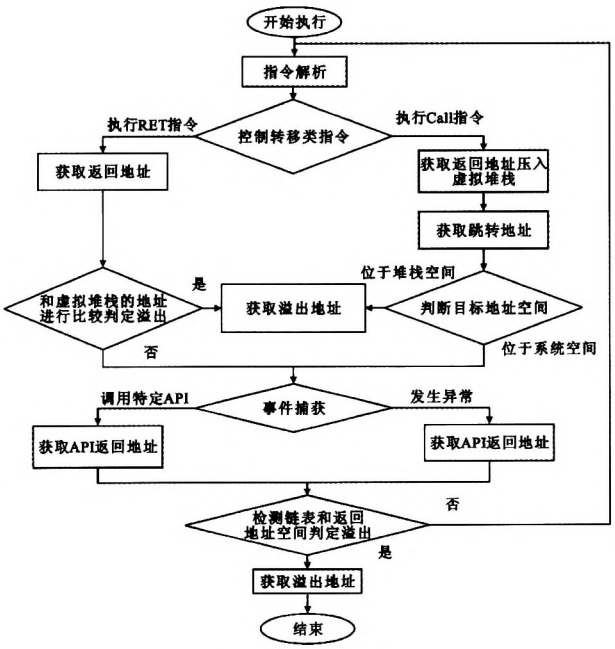

图3 溢出检测流程

(3)函数调用序列流程分析

程序的整个执行流程都通过函数调用来实现,而所有的函数调用都是基于Call指令来完成,因此通过插桩CALL指令,在插入的装代码中记录程序的指令调用流程和函数传递的参数及返回值等信息来记录程序的执行流程。函数调用流程记录过程如下图4所示:

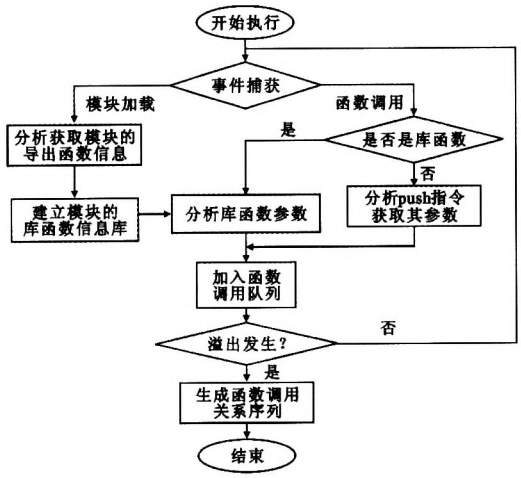

图4 函数调用序列记录流程

系统采用循环队列的方式记录程序调用流程,总共记录了200次的函数调用,到溢出发生时,队列中记录的时溢出发生前最近的200次函数调用关系序列图,溢出发生后系统将根据记录信息生成函数调用流程关系图。

2 系统有效性分析

为了验证系统的有效性,对IE、office、以及Adobe等软件的多个漏洞样本进行了测试验证,下面就具体以CVE-2012-0158漏洞进行说明。测试环境为Windows XP3、office2010[9]。

CVE-2012-0158是一个winword的溢出漏洞,winword的COMMCTL.OCX组件中由于逻辑错误导致溢出漏洞发生。图5是将metasploit生成的样本经过该系统测试后得出的结果,测试结果与实际分析结果吻合。

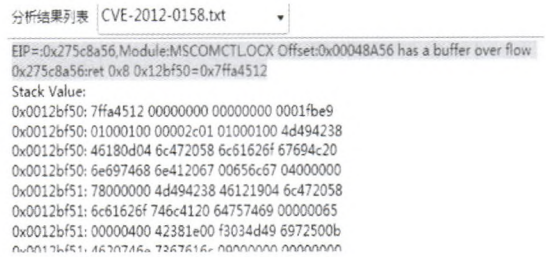

图5 系统分析结果

图6是系统记录的溢出发生前的函数调用序列(这里只列出了4层,红色标记的一层为溢出发生所在的函数):

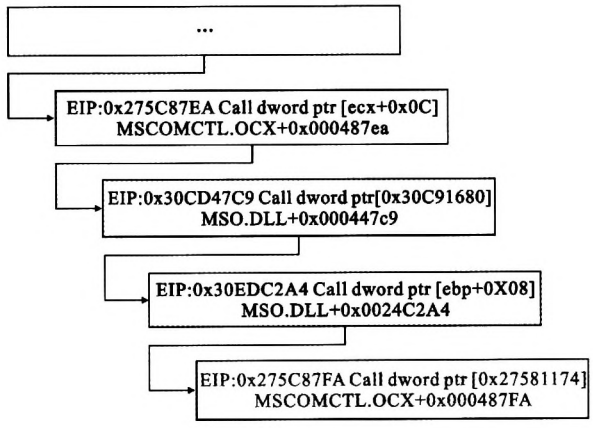

图6 函数调用流程图

表1是其他样本测试结果,测试样本均来自metasploit生成,限于篇幅这里只列出了一部分,测试发现系统没有发现误报,能够准确检测到缓冲区溢出并定位到溢出发生的位置。

(下转第87页)

1162-1167.

[3] BERESOFRD A R, RICE A, SKEHIN N, et al. MockDroid: Trading Privacy for Application Functionality on Smartphones [C]//Proc. of the 12th Workshop on Mobile Computing Systems and Applications. Phoenix, USA, 2011.

[4] ZHOU YAJIN, JIANG XUXIAN. Dissecting Android Malware: Characterization and Evolution [C]//2012 IEEE Symposium on Security and Privacy.2012:95-109.

[5] JACOB G, DEBAR H, FILIOL E. Behavioral detection of malware: from a survey towards an established taxonomy [J]. Journal in computer Virology, 2008, 4(3): 251-266.

[6] 贾菲,刘威.基于 Android 平台恶意代码逆向分析技术的研究[J].信息网络安全, 2012(4):61-63, 84.

[7] 杨广亮,龚晓锐,姚刚等.一个面向 Android 的隐私泄露检测系统[J].计算机工程,2012,38(23):1-6.

[8] MICHAEL GRACE, WU ZHOU, XUXIAN JIANG and AHMAD-REZE SADEGHI: Unsafe Exposure Analysis of Mobile In-App Advertisements [C]// the 5th ACM Conference on Security and Privacy in Wireless and Mobile Networks. 2012.

[9] 冯博,戴航,慕德俊等.Android 恶意软件检测方法研究[J].计算机技术与发展, 2014 (2): 149-152.

[10] Android.Manifest.permission[EB/OL]. http://developer.android.com/reference/android/Manifest.permission.html.

作者简介:

楼赟程(1989—),男,硕士,主要研究方向为 Android 手机安全;

施　勇(1979—),男,博士,讲师,主要研究方向为网络与信息安全;

薛　质(1971—),男,博士,教授,主要研究方向为网络与信息安全。■

(上接第 82 页)

表 1　系统测试结果

编号	测试软件	溢出类型	溢出模块	实际符合度
CVE20080015	IE6	栈溢出	mshtml.dll	√
CVE20100806	IE7	释放重后引用	iepeers.dll	√
CVE20103333	Office2003	栈溢出	mso.dll	√
CVE20130417	Adobe94	堆溢出	Acrord32.dll	√

3　结语

实验表明该系统的检测结果与漏洞实际分析结果符合,说明动态插桩技术结合漏洞利用特征来进行缓冲区溢出漏洞检测的正确性和有效性,从而快速对漏洞样本进行检测并辅助进行对漏洞原理分析。由于采用动态插桩会影响程序执行效率,未来会插桩策略上进一步优化,以提高程序分析效率。

参考文献:

[1] 王清,张东辉,周浩.0da 安全:软件漏洞分析技术[M].北京:北京电子工业出版社,2011:267-274.

[2] Intel.Pin-A Dynamic Binary Instrumentation Tool[EB/OL]. (2012-07-13)[2014-11-28]. https://software.intel.com/en-us/articles/pintool/wbia09.pdf.

[3] 高迎春,周安民,Windows DEP 数据执行保护技术研究[J].信息安全与通信保密,2013(09):77-80.

[4] 代伟,刘智,刘益和.基于二进制的动态污点分析[J].计算机应用研究,2014(08):2498-2505.

[5] 诸葛建伟,陈丽波,田凡.基于类型的动态污点分析技术[J].清华大学学报(自然科学版),2012-10-23[10]:1321-1328.

[6] 陈丹."二进制审核"方式的缓冲区溢出漏洞挖掘[J].办公自动化应用,2014(09):43-45.

[7] 段钢.加密与解密[M].北京:电子工业出版社,2008:306-311.

[8] 黄志军,郑涛.一种基于 DBI 的 ROP 攻击检测[J].计算机科学,2012(09):120-124.

[9] 启明星辰.CVE-2012-0158 分析报告[EB/OL].(2012-04-28)[2014-11-28].http:// http://www.venustech.com.cn/NewsInfo/449/13620.Html.

作者简介:

刘露平(1988—),男,硕士研究生,主要研究方向:信息安全、网络通信;

方　勇(1966—),男,博士,教授,主要研究方向:网络与信息系统安全;

刘　亮(1982—),男,博士,讲师,主要研究方向:网络与信息系统安全;

龙　刚(1986—-),男,硕士,主要研究方向:信息安全、网络通信。■

万方数据