一种远程缓冲区溢出漏洞检测模型及系统实现*)

许俊杰　蔡皖东

(西北工业大学计算机学院　西安 710072)

摘　要　操作系统和应用软件中的潜在远程缓冲区溢出漏洞是信息系统面临的最严重安全威胁之一。检测软件中潜在的远程缓冲区溢出漏洞对于提高信息系统安全性具有重要的意义。本文采用面向二进制代码的动态分析方法，提出了基于错误注入技术的远程缓冲区溢出漏洞检测模型，介绍了系统结构和模块功能，详述了系统中关键技术，给出了系统测试结果。

关键词　错误注入，缓冲区溢出，漏洞检测

A Model for Detecting Remote Buffer Overflow Vulnerabilities and its Implementation

XU Jun-jie　CAI Wan-dong

(College of Computer Science, Northwestern Polytechnical University, Xi'an 710072 ,China)

Abstract　Remote buffer overflow vulnerabilities existing in operating system and application software constitute one of the most serious security threat towards computer network and information on it. Detecting potential remote buffer overflow vulnerabilities in softwares is of great significancy to the improvement of information system security. This paper adopts a method of binary-code oriented dynamical analysis, introduces a SWIFI based model for detecting remote buffer overflow vulnerabilities, explains the system structure and module functionality, dilates on the key technologies employed and shows the result of the system test.

Keywords　Fault injection, Buffer overflow, Vulnerability detection

1　引言

随着计算机网络技术的高速发展，人们的生产生活对网络的依赖程度越来越高。与此同时，网络犯罪活动也越来越猖獗。存在于操作系统和应用软件中的远程缓冲区溢出漏洞为网络攻击者大开了方便之门。

缓冲区溢出漏洞是一种软件中边界条件、函数指针等设计不当造成的地址空间错误。攻击者通过向一个有限空间的缓冲区中拷贝过长的字符串达到运行恶意代码的目的[1]。缓冲区溢出漏洞普遍存在于包括 Windows、Unix、Linux、Solaris 和 Mac OS 在内的各种主流操作系统和运行在这些操作系统上的软件当中。据统计，缓冲区溢出漏洞占所有软件漏洞的 40%～50%[2]。能够通过网络远程利用的缓冲区溢出漏洞称为远程缓冲区溢出漏洞。一次对远程缓冲区溢出漏洞成功的攻击可以使运行存在漏洞的软件的 Internet 主机或服务器被攻击者完全控制。

远程缓冲区溢出漏洞存在之普遍，危害之严重使其成为当今软件和信息系统安全的重大威胁，关系到企业生存和国家安全。网络攻击行为已经从以往的以炫耀技术为主转向以获取经济利益为目的。一旦攻击者发现某个广泛使用的软件中的远程缓冲区溢出漏洞并对其发动攻击，就会给开发和使用该软件的企业和个人带来巨大的损失。发现软件系统中的远程缓冲区溢出漏洞对于测评软件的安全性、及时开发安全补丁、防范网络安全风险具有非常重要的意义。

2　缓冲区溢出漏洞检测技术研究现状

国内外对缓冲区溢出检测技术进行了大量的研究工作。从检测方法上，可以分成基于源代码的静态检测、基于源代码的动态检测、基于目标代码的静态检测和基于目标代码的动态检测等。

基于源代码的静态检测技术通过对源代码的扫描和分析，对软件漏洞发生的模式进行识别，进而实现对缓冲区溢出漏洞的检测[5]。这种检测技术的局限性在于误报率较高而且需要提供源代码，而大量的商业软件或共享软件都不提供源代码。

基于源代码的动态检测技术主要通过对源代码执行时的程序内存访问情况进行监控，以此检测和发现缓冲区溢出漏洞[6]。这种检测技术同样需要提供源代码，无法对二进制代码进行检测。

基于目标代码的静态检测技术主要采用逆向工程技术对目标代码进行反汇编分析，然后采用类似于源代码静态检测技术对代码中可能存在的缓冲区溢出漏洞进行检测[7]。这种检测技术的缺点是反汇编工作量大、技术要求高、误报率高。另外，逆向工程可能受到合同或法规的制约。

基于目标代码的动态检测技术是近年来提出的一种缓冲区溢出漏洞检测技术，具有较大的发展前景。它采用类似于软件测试中的黑箱穿透测试技术对目标代码进行动态测试，模拟缓冲区溢出攻击，将攻击代码注入被测目标代码中，以检测是否存在缓冲区溢出漏洞。这种技术的缺点是攻击代码编

*)西北工业大学研究生创业种子基金资助(项目编号：Z200758)。**许俊杰**　硕士研究生，主要研究方向为网络信息安全；**蔡皖东**　教授，博士生导师。

写难度高，检测效率低。

采用基于目标代码的动态检测技术检测远程缓冲区溢出漏洞，优点在于不需要源代码、无误报，需要解决的问题是如何提高检测效率。

3 基于错误注入的远程缓冲区溢出漏洞检测模型

3.1 缓冲区溢出攻击原理

针对缓冲区溢出漏洞的攻击行为称为缓冲区溢出攻击。最常见的两种缓冲区溢出攻击类型是栈溢出攻击和堆溢出攻击。下面以 Windows 系统为例来介绍这两种缓冲区溢出攻击原理。

3.1.1 栈溢出攻击

当一个函数调用另一个函数时，其线程栈的结构如图 1 所示。在正常情况下，被调函数完成后被调函数返回地址弹出到 EIP 寄存器，然后从 EIP 寄存器指向的地址取指令继续主调函数的执行。当发生缓冲区溢出攻击时，过长的数据被拷贝到被调函数局部变量的缓冲区中，并沿着内存地址增长的方向覆盖掉被调函数返回地址。成功的缓冲区溢出攻击会用精心设计的数据覆盖被调函数返回地址，使程序流程直接或间接转向攻击代码(shellcode)。当被调函数结束时，已经被篡改过的被调函数返回地址弹出到 EIP 寄存器，开始攻击代码的执行。

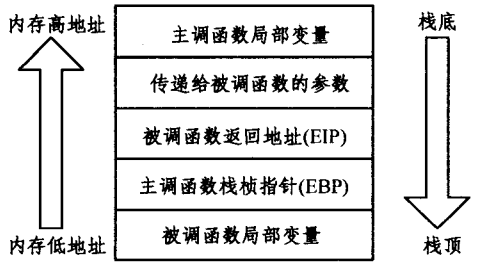

图 1 函数调用时的栈结构

3.1.2 堆溢出攻击

Windows 堆管理器采用双向链表的形式管理未分配的内存块。当遇到分配内存请求时，堆管理器选择一块未分配的内存块从未分配内存块双向链表中摘除，将其地址返回。摘除时用到指令 mov [ecx], eax，其中 ECX 寄存器存放的是指向前一个未分配内存块的指针，EAX 寄存器存放的是指向下一个未分配内存块的指针。如图 2 所示，发生堆溢出攻击时，过长的数据被拷贝到已分配的内存，并沿着内存地址增长的方向覆盖掉未分配内存块的前向指针和后向指针。当堆管理器选择被破坏了的未分配内存块进行分配时，mov [ecx], eax 会把前向指针的内容拷贝到后向指针指向的地址。攻击者可以用攻击代码的地址覆盖前向指针，用异常处理函数入口地址的地址覆盖后向指针，然后触发异常，达到执行攻击代码的目的。

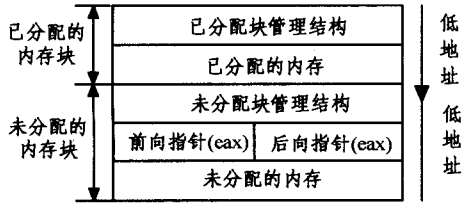

图 2 堆的结构

远程缓冲区溢出攻击是针对可以远程利用的缓冲区溢出漏洞进行的攻击。与一般的缓冲区溢出攻击不同的是，用于远程缓冲区溢出攻击中溢出的数据是通过网络远程输入的。

3.2 远程缓冲区溢出漏洞检测模型

远程缓冲区溢出攻击能够成功的必要条件之一是：用来溢出缓冲区的数据必须经过精心设计。被调函数返回地址或未分配内存块前向指针和后向指针被垃圾值覆盖会引发内存访问冲突或者非法指令异常。如果有调试器附加到发生异常的进程，异常会被报告给调试器，由调试器决定如何处理；如果没有调试器附加到进程且进程自身无法处理发生的异常，异常就会被报告给操作系统，由操作系统结束进程。

通过对被检测的进程进行网络层错误注入来模拟不成功的远程缓冲区溢出攻击，同时附加调试器到被检测的进程来捕获异常，根据捕获到的异常判断是否发生缓冲区溢出。一旦确定被检测进程发生了缓冲区溢出且缓冲区溢出事件与网络层错误注入存在因果关系，就可以断定被检测的软件存在远程缓冲区溢出漏洞。

基于错误注入技术[3]的远程缓冲区溢出漏洞检测模型将错误注入技术和缓冲区溢出检测技术相结合，能够主动检测 Windows 平台下软件系统中潜在的远程缓冲区溢出漏洞。该模型具有针对性强、不需要源代码、无误报等优点，可以应用在软件测试、安全性测评等领域。

4 系统实现

本系统由检测代理、错误注入器和控制台等三个基本模块组成，模块之间的关系如图 3 所示。

检测代理和被检测对象运行在同一台主机或服务器上，负责设置运行环境，检测并向控制台报告发生在被检测软件系统中的缓冲区溢出事件及相关信息。

错误注入器内部包含协议分析器，脚本生成器和注入引擎三个主要组件，其中协议分析器接收检测代理捕获到的网络数据流，分析其通信协议特征，确定注入点，写入协议知识库；脚本生成器根据协议知识库确定注入点，根据漏洞知识库确定注入内容，生成错误注入脚本，写入脚本库；注入引擎根据脚本库里的错误注入脚本完成错误注入。

控制台负责控制检测代理和错误注入器，实现对漏洞检测过程的管理和漏洞库管理。

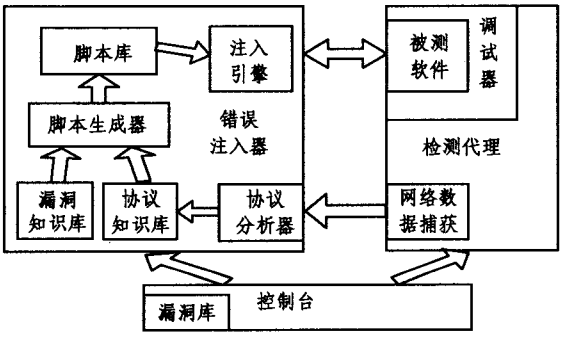

图 3 系统模块关系示意图

4.2 关键技术

4.2.1 缓冲区溢出事件检测

在程序运行过程中，如果发生异常事件，则向调试器报告。使用 Win32 调试 API 可以捕获这些异常事件。检测代

理通过带 DEBUG_ONLY_THIS_PROCESS 标志的 CreateProcess 函数或者 DebugActiveProcess 函数获得对被检测进程的调试权限，然后调用 WaitForDebugEvent 函数等待调试事件。被调试的进程试图访问不可访问的内存，遇到非法指令或者发生其它异常都会产生 EXCEPTION_DEBUG_EVENT 事件。检测代理检测到 EXCEPTION_DEBUG_EVENT 事件后通过分析异常类型、线程上下文、调用栈和内存状态等与事件相关的信息判断是否发生了缓冲区溢出。

4.2.1.1 堆溢出检测原理

页堆管理器(Pageheap.exe)在应用程序和系统核心之间引入了软件验证层，该层验证包括内存分配和释放在内的所有动态内存操作。它启用 Ntdll.dll 系统库中现有的验证层，发现错误后将错误信息报告调试器。Pageheap.exe 有两种检测模式：正常页堆和整页堆。“正常页堆”在分配的内存块前后写入特定的标记，释放内存时通过检查标记是否被破坏来判断是否发生了溢出。“整页堆”处于分配的内存块结尾的位置，防止一个不可访问的页。当溢出发生，进程试图读写超过已分配内存结尾的地址空间时就会触发访问冲突异常(EXCEPTION_ACCESS_VIOLATION)。和正常页堆只有内存释放时才有机会检测到溢出相比，整页堆可以实现堆溢出的实时检测，故本模型采用整页堆检测模式。

4.2.1.2 栈溢出检测原理

栈溢出的基本原理是通过覆盖栈里的 EIP 指针来改变程序流程，执行 shellcode。要达到改变程序流程的目的，溢出字符串的内容必须经过精心设计。如果覆盖 EIP 的只是一般的字符串，则可能导致 EIP 指向的内存不可读，取指令时触发访问冲突异常(EXCEPTION_ACCESS_VIOLATION)或者 EIP 指向的内存存储的不是合法的指令，触发非法指令异常(EXCEPTION_ILLEGAL_INSTRUCTION)。结合错误注入，在发生访问冲突异常或者非法指令异常时，通过比较 EIP 内容和注入字符串的内容就可以判断出是否发生栈溢出和定位溢出点。

4.2.2 错误注入技术

错误注入技术最早是用硬件实现的，可以模拟系统中的硬件失败，被应用于硬件系统的可靠性测试。人们很快发现软件实现的错误注入(SWIFI，Software Implemented Fault Injection)技术可以用来测试软件系统的可靠性。

SWIFI 可以分为编译时注入和运行时注入两类，其中运行时注入不需要被检测软件系统的源代码。运行时注入 SWIFI 的注入位置通常包括内存空间、系统调用层和网络层。本系统通过网络层的运行时 SWIFI 技术实现远程缓冲区溢出漏洞检测。

软件中的远程缓冲区溢出漏洞产生的原因是使用网络输入的字符串或其它长度不固定的二进制数据之前没有做长度和有效性检查，据此可以确定网络层错误注入的注入点为网络数据流中的字符串和其它长度可变数据。使用公开的通信协议的软件很容易确定注入点，对于使用私有通信协议的软件，需要使用模式识别技术分析被检测软件的网络数据流，确定错误注入点。注入点确定后，结合漏洞知识库生成注入脚本，完成错误注入。

5 系统测试

为检验模型的可用性，对本模型的检测能力做了测试。被检测的程序包括两个专为系统测试编写的带远程缓冲区溢出漏洞的程序、一个存在已公开的远程缓冲区溢出漏洞的程序和一个事先不知道是否有漏洞的程序。检测结果见表 1。

表 1 测试结果

程序名称	漏洞类型	来源	是否检测到漏洞
有栈溢出漏洞的测试目标	存在栈溢出漏洞	手工编写	是
有堆溢出漏洞的测试目标	存在堆溢出漏洞	手工编写	是
UltraVNC 1.0.1	存在已公开的漏洞(CVE-2006-1652)	http://sourceforge.net/projects/ultravnc/	是
某 FTP 客户端软件	未知是否存在漏洞	开发者网站	是

结束语 和其它缓冲区溢出漏洞检测方法相比，本文提出的检测模型专门用于检测远程缓冲区溢出漏洞，具有无误报、不需要源码等优点。系统测试和实际应用表明，基于错误注入技术的远程缓冲区溢出漏洞检测系统能够检测到专为系统测试编写的远程缓冲区溢出漏洞，也能够检测出商业软件中已知和未知的漏洞。

参 考 文 献

[1] 王炜，方勇. 缓冲区溢出教程. 北京：北京中电电子出版社，2005

[2] Wagner D, Foster J, Brewer E, et al. A first step towards automated detection of buffer overrun vulnerabilities// Network and Distributed System Security Symposium. San Diego, CA, February 2000

[3] Hsueh M, Tsai T, Iyer R. Fault injection techniques and tools. IEEE Computer, April 1997:75-82

[4] Wilander J, Kamkar M. A Comparison of Publicly Available Tools for Dynamic Buffer Overflow Prevention// Proceedings of the 10th Network and Distributed System Security Symposium. San Diego, CA, February 2003:149-162

[5] Viega J, Bloch J T, Kosho T, et al. ITS4: A Static Vulnerability Scanner for C and C++ Code// Annual Computer Security Applications Conference. December 2000

[6] Baratloo A, Singh N, Tsai T. Transparent runtime defense against stack smashing attacks// Proceedings of the 2000 USENIX Annual Technical Conference. San Jose, CA, June 2000: 251-262

[7] Duraes J, Madeira H. A Methodology for the Automated Identification of Buffer Overflow Vulnerabilities in Executable Software Without Source-Code// Proceeding of the LADC 2005, October 2005