doi :10.3969/j.issn.1671-1122.2009.05.017

基于漏洞特征和 Fuzz 技术的 Windows 漏洞挖掘模型研究

徐俊扬 ， 文伟平

（北京大学软件与微电子学院信息安全系，北京 102600）

摘 要：Windows漏洞挖掘已经成为网络安全技术一个重要的组成部分。在结合逆向工程和fuzz技术的Windows漏洞挖掘模型中加入了漏洞特征的因素：本文首先介绍了逆向工程、漏洞特征建模和fuzz技术的概念和研究内容，接着介绍了基于漏洞特征的Windows漏洞挖掘模型原理。

关键词：逆向工程；漏洞特征；Fuzz技术；漏洞挖掘

中图分类号：TP393.08 文献标识码：A

Vulnerability mining model for Windows based on vulnerability characteristics and Fuzz technology

XU Jun-yang, WEN Wei-ping

(Department of Information Security, SSM, Peking University, Beijing 102600, *China)*

Abstract: Windows vulnerability mining has become an important part of the network security technology. We consider the vulnerability characteristics in the Windows vulnerability mining model based on reverse engineering and technology fuzz technology: first we introduce the reverse engineering and vulnerability characteristics, and then introduce the principle of vulnerability mining model for Windows based on vulnerability characteristics.

Key words: Reverse engineering; Vulnerability characteristics; Fuzz technology; Vulnerability mining

0 引言

随着网络和计算机的普及，Windows 操作系统在全球范围内更加流行。以此同时，基于该系统的漏洞层出不穷，特别是 Windows 每月公布的安全公告里的漏洞都有可能使自己的计算机被别人所控制。因此，人们开始对 Windows 软件漏洞重视起来，Windows 系统漏洞挖掘技术也就成为了网络安全技术的一个重要组成部分。

软件漏洞挖掘根据分析源的不同，可以分为开源软件和非开源软件的漏洞挖掘。开源软件的漏洞挖掘是指在获得目标软件源代码的基础上，对代码进行分析而发现漏洞，其效率与准确性比非开源软件好很多。而 Windows 本身和基于 Windows 的大多数软件都不是开放源代码的，有效地对其进行漏洞挖掘比较困难。逆向工程分析和 Fuzz 都是非开源软件的漏洞挖掘技术的主要方法。

通过逆向工程，能够很好的了解目标软件的结构和运行流程，从而能够比较精确的定位到漏洞代码的位置，同时利用 Fuzz 技术能够最大程度上实现漏洞挖掘的自动化，因此结合逆向工程和 Fuzz 技术的 Windows 漏洞挖掘模型 [1] 一定程度上提高了非开源软件的漏洞挖掘效率和准确性。但是现有的逆向工程只是能够定位一些轻量级的简单编程错误的代码位置，而忽略了一些逻辑上可能导致漏洞存在的代码段，在漏洞挖掘中不能做到通用性。因此研究的目的就是在结合逆向工程和 Fuzz 技术的 Windows 漏洞挖掘模型中考虑漏洞特征，以提高该模型的通用性。.

1 逆向工程技术与漏洞特征建模

1.1 逆向工程技术

术语“逆向工程”(Reverse Engineering)来自于硬件领域，是通过检查样品来开发复杂硬件系统规约的过程 [2]。主要为了商业或军事利益而分析研究他人的硬件系统，发现其工作原理，以达到复制硬件系统的目的。

随着软件业的发展，逆向工程这一术语被引入了软件工程领域，逆向工程可以用于揭示已有系统工作原理的过程，或者是用于描述创建现有文档的联机文档的过程等应用 [3]。

逆向工程技术主要分为静态分析和动态分析两大类。其中，静态分析技术主要包括：有向图分析、污点数据传播分析、IDC 脚本分析和整数限制分析；动态分析技术主要包括：格式分析、黑盒测试和虚拟堆栈分析 [1]。根据漏洞挖掘的需要，对软件的逆向工程最终都需要通过特定的逆向平台对软件程序进行反汇编，得到相应地汇编代码，然后从汇编代码中找到可能出错函数或存在逻辑错误的代码段的位置。

1.2 漏洞特征建模

在现有的Windows漏洞中，有许多是因为编写操作系统的DLL库文件的语言本身的一些特性导致，也有一些是因为代码段本身存在逻辑问题导致的。

现有的Windows漏洞中很大一部分是由语言本身特性导致的缓冲区溢出漏洞，引起缓冲区溢出问题的根本原因是软件程序中使用了不带长度检查的字符串类处理函数(C函数)或是向缓冲区拷贝数据时没有进行边界检查。C的编译器一般不对数组和指针的引用进行边界检查，使得在程序中可以通过数组和指针的引用来对内存数据区中任何数据进行访问，而不仅仅是程序中定义的区域。标准C库中存在许多不安全字符串操作函数。表1对可能导致缓冲区溢出漏洞的不安全字符串处理函数作了一个基本总结[4]。

函数名	不安全性
strcpy、wcscpy、lstrcpy、_tcscpy、_mbscpy	strcpy、wcscpy、lstrcpy、_tcscpy、_mbscpy 这些字符串拷贝函数不检查目标缓冲区的大小，也不检查null或其他无效指针。
strcat、wcscat、lstrcat、_tcscat、_mbscat	不能保证这些函数会用null来结束目标缓冲区，也不会检查null或其他无效指针。
strncpy、wcsncpy、_lstrncpy、_tcsncpy、_mbsncpy	不能保证这些函数会用null来结束目标缓冲区，也不会检查null或其他无效指针。
memcpy、CopyMemory	目标缓冲区必须足够大，否则会发生缓冲区溢出。
printf家族包括printf、_sprintf等)	可能导致格式化缓冲区溢出漏洞。
sprintf、swprintf	不能保证函数会用null结束目标缓冲区。
_snprintf、_swnprintf	不能保证函数会用null结束目标缓冲区。
gets	不检查拷贝的缓冲区大小。
C++标准模板库中的流操作符 >>	不检查拷贝的目标缓冲区大小。

表1 不安全字符串处理函数表

一旦在程序中使用了这些不安全的字符串处理函数，并在不进行严格边界检查的情况下，就会导致缓冲区溢出漏洞的产生。例如典型的ms06-040就是基于这种情况产生的缓冲区溢出漏洞。

另一类Windows漏洞是由于代码段本身存在的逻辑问题导致，例如IE7.0存在的ms09-002漏洞。Internet Explorer的CFunctionPointer函数没有正确地处理文档对象，如果以特定序列附加并删除了对象，就可以触发内存破坏。该弱点实际存在于mshtml.dll中。CFunctionPointer对象在其构造函数中没有正确地引用文档对象，导致该文档对象可能在CFunctionPointer对下释放前被释放，而CFunctionPointer会继续使用这个已经被销毁的文档对象。因此代码段逻辑也可能导致漏洞的产生。

从上面的分析可以得出Windows漏洞特征：调用不安全的字符串操作函数导致缓冲区溢出与代码段逻辑错误导致某些不可预测的漏洞。对于由字符串操作函数导致的漏洞，其特征比较容易提取，主要是把这些函数的函数名当作关键字，用于在反汇编代码中进行匹配。对于由代码段逻辑错误导致的漏洞，其特征提取比较困难，需要对其进行建模，构建出可能导致漏洞的一小块代码，然后在反汇编代码中进行匹配。

因此在模型中，我们首先利用ClearBug写出基于已经构建出的漏洞特征模型的IDC脚本[5]，然后主要利用IDA Pro进行反汇编，将二进制代码还原成汇编代码，通过静态分析了解程序的功能结构和运行流程。同时用ClearBug查找出可能出现漏洞的代码位置。最后利用Windbg进行动态调试，分析该代码位置是否存在漏洞。

2 Fuzz技术

Fuzz又称模糊测试，是一种用来发现软件漏洞的测试技术，其思想就是利用“暴力”来实现对目标程序的自动化测试，然后监视检查其最后的结果，如果符合某种情况就认为程序可能存在某种漏洞或者问题。与传统的测试技术不同，Fuzz测试不是完善目标软件的完整性和正确性，相反，Fuzz技术是通过向软件外部接口发送一些随机数据，致使目标软件发生异常，从而发现漏洞。因此采用Fuzz技术能一定程度上使得漏洞挖掘自动化了。

在模型中，我们主要是针对Windows API进行Fuzz测试，这些API主要是Windows下的DLL库文件导出的一系列函数。DLL库文件是一种将某种功能的具体实现封装起来，然后只向外提供一个简单方便的可调用函数。API Fuzz测试是向DLL库文件导出的函数传送一些意外数据，然后监测可能发生打的异常。然而，没有考虑函数间依赖关系的API Fuzz测试通常会产生一些错误，这是因为需要被调用的函数没有在目标函数被调用之前调用，意外数据将不能达到目标函数的代码段[6]。

例如，为了对ReadFile( )进行API Fuzz测试，我们首先找到该函数的原型，其原型如下；然后直接调用该函数，向其传入一些意外数据；最后执行报错。这是因为在调用ReadFile( )函数之前必须调用CreateFile( )函数。

```
BOOL WINAPI ReadFie(
HANDLE hFile,
LPVOID lpBuffer,
DWORD nNumberOfBytesToRead,
LPDWORD lpNumberOfBytesRead,
LPOVERLAPPED lpOverlapped );
```

因此我们在进行API Fuzz测试前，必须弄清楚与目标函数具有依赖关系的函数。由同一个dll文件导出的API函数通常是相关联的，因为一个dll文件把功能相似的API函数放在一起。因此，处于同一个dll文件中的API函数具有依赖关系。在模型中，我们打算基于WinDbg extension[7]插件实现该功能，WinDbg extension是一个动态链接库，由WinDbg支持的用来实现一些附加功能。当一个软件正在运行时，该插件能够跟踪同一个DLL库文件中的函数调用，并且按调用顺序把这些函数写入日志文件里。

3 基于漏洞特征的Windows漏洞挖掘模型

鉴于逆向工程和Fuzz技术各存在优缺点，在结合二者优点的基础上，提出的结合逆向工程和Fuzz技术的Windows漏洞挖掘模型在一定程度上将自动化和目的性结合起来[1]。我们的模型是基于该模型，在逆向工程过程中加入了漏洞特征建模，使得该改模型具有通用性。

以发掘Windows下的DLL库文件中可能存在的缓冲区溢出漏洞为例，介绍一下这套模型的具体流程。在对DLL库文件进行逆向分析和Fuzz测试前，我们必须先对漏洞特征进行建模。首先需要找出可能导致缓冲区溢出的字符串操作函数，然后把这些函数进行漏洞特征建模，也就是把这些函数的函数名作为关键字或者把这些函数对应的一段汇编代码作为匹配代码段。最后，根据漏洞特征模型，编写基于IDA的IDC脚本文件，以备对反汇编后的程序进行关键位置扫描。其流程如图1所示。

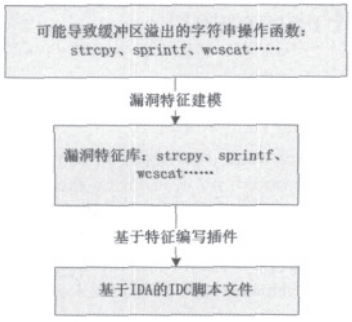

图1 漏洞特征建模流程图

在漏洞特征模型的基础上，我们将对DLL库文件进行缓冲区溢出漏洞挖掘，其具体流程如图2所示。首先，将目标软件通过IDA Pro获得反汇编代码，然后利用ClearBug中已经写好的Idc脚本文件在代码中搜索strcpy、sprintf等容易发生缓冲区溢出的关键函数，并进一步分析出调用该字符串操作函数的上一层函数PreFunction。此时，如果进一步手工分析函数之间的依赖关系，对程序流程进行逆向跟踪，分析程序是否对变量边界进行严格判断，这样跟踪的人工成本很高，可能要回溯很长一段才会到程序对变量初始化的位置，或是直接对PreFunction函数进行API Fuzz而不考虑函数间的依赖关系，则有可能因这种依赖关系产生错误从而不能到达调到关键字符串操作函数的代码处。这时，我们可以利用Windbg extension插件跟踪PreFunction与DLL库文件中其他API之间的依赖关系，然后对适当的API进行Fuzz测试，使其最终能够达到关键字符串操作函数的代码处。根据选定的适当API，编写测试程序对其进行Fuzz测试，如果API出现异常（比如异常退出、死循环锁定），就可以判断目标软件可能存在缓冲区溢出漏洞。

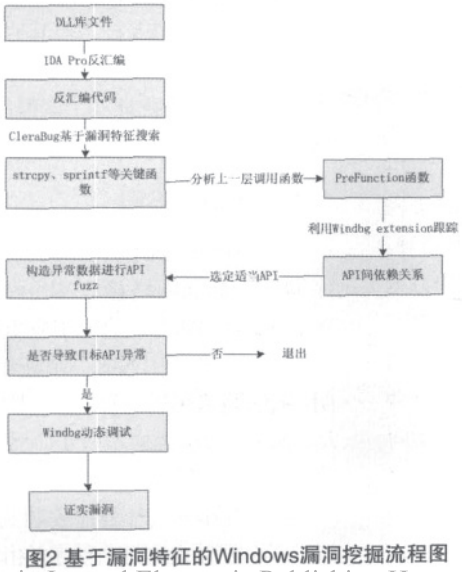

图2 基于漏洞特征的Windows漏洞挖掘流程图

4 结论

基于人们在Windows漏洞挖掘方面所做的努力，越来越多的基于缓冲区溢出的漏洞被发现，这也意味着基于缓冲区溢出的Windows漏洞将越来越少，现有的一些致力于缓冲区溢出漏洞挖掘的工具将越来越困难地发现此类漏洞。本文的创新点是在现有的将逆向工程和Fuzz技术结合的Windows漏洞挖掘模型中，考虑了漏洞特征，这样不仅能够发现一些轻量级的缓冲区溢出漏洞，也可以发现一些由于代码逻辑出错的漏洞。这就增强了Windows漏洞挖掘模型的通用性。

但是，该模型仍有很多不足之处。其难点之一就是对于漏洞特征进行建模。它只能从已有Windows漏洞中提取出逻辑错误特征，并进行建模，而这种建模是基于汇编代码段，存在一定的难度与准确性。其次，并不能对未知的逻辑错误进行漏洞特征建模。最后，对找到可能存在漏洞的API进行Fuzz测试时，完全弄清楚其与其他关联API之间的依赖关系也有一定得难度。因此，在今后的研究工作中，主要是针对“Windows漏洞特征建模”以及DLL库文件中的“API之间的依赖关系”进行进一步研究。 （责编 杨晨）

参考文献：
[1] 刘坤.结合逆向工程和fuzz技术的Windows软件漏洞挖掘模型研究[D]. 成都：四川大学，2008
[2] M.Rekoff.On reverse engineering.IEEE Transactions on Systems, Man and Cybernetics. 2/4:244-252
[3] Scott R. Tilley, Michael J. Whitney, Hausi A .Muller,et al.Personalized information structures.In Proceedings of the 11th Annual International Conference on Systems Document (SIGDOC 93). Pages325-337, ACM (Order Number 6139930),October 1993
[4] 曾颖. 基于补丁比对的Windows下缓冲区溢出漏洞挖掘技术研究[D]. 解放军信息工程大学，2007
[5] 刘波. ClearBug VS BugScam[D]. 北京：北京大学软件与微电子学院信息安全系，2008.
[6] Choi YoungHan, Kim HyoungChun, Oh HyungGeun, Lee Dohoon. Call-Flow Aware API Fuzz Testing for Security of Windows Systems. Electronics Telecommunications Research Institute (ETRI), 2008.
[7] Debugging Tools for Windows-Overview. http://www.microsoft.com/whdc/devtoools/debugging/default.mspx.

作者简介：徐俊扬（1986 —），男，硕士研究生，研究方向：信息网络安全；文伟平（1976 —），男，副教授，主要研究方向：网络攻击防范、恶意代码研究、信息系统逆向工程和可信计算技术等。