基于栈的恶意程序隐式系统调用的检测方法*)

李毅超　何子昂　曹　跃
(电子科技大学计算机科学与工程学院　成都 610054)

摘　要　提出了一种检测恶意程序中隐式系统调用的方法。该方法使用地址栈和地址栈图来检测恶意程序中隐式的系统调用信息，其中，地址栈将每个栈的元素和栈操作的指令相结合，而地址栈图抽象地表示可执行体并且检测恶意的系统调用。通过实验表明，这是一种有效的方法。

关键词　恶意程序，隐式调用，地址栈，地址栈图

A Stack-related Method for Detecting Obfuscated System Calls of Malware

LI Yi-Chao　HE Zi-Ang　CAO Yue
(School of Computer Science & Engineering, UEST of China, Chengdu 610054)

Abstract　This paper presents a method to detect obfuscated system calls of malware. The idea is to use address stack and address stack graph to detect obfuscated system calls of malware. An address stack is used to associate each element in the stack to the instruction that pushes the element. An address stack graph may be created by abstract interpretation of the binary executable and may be used to detect obfuscated calls. The experiment proves the method is effective.

Keywords　Malware, Obfuscated calls, Address stack, Address stack graph

1　引言

目前，病毒、蠕虫、木马、间谍软件以及后门等恶意程序大都使用了自迷惑技术。自迷惑技术是程序自身在不改变原程序语义的情况下通过代码变换来对抗逆向工程的一种技术。具有自迷惑技术的恶意程序被反病毒组织定义为多态恶意程序，它们在感染下一个宿主之前实施迷惑[1]。进行反复地迷惑使得两个来自于同一恶意程序样本的副本完全不同，看不出任何继承关系，还更加难以进行逆向工程。但是恶意程序为了达到特定目的须使用文件和网络等系统资源。基于对上千恶意程序的分析，安全公司积累了一系列恶意程序显式的系统调用序列。一些病毒扫描器通过显式的系统调用序列来确定恶意程序，而使用隐式系统调用的恶意程序则能安全通过这种扫描器。隐式系统调用是将系统调用指令(call)用其它语义相同而语法不同的组合指令替换的调用。

本文对栈相关指令的迷惑技术提出了一种隐式系统调用的静态检测方法。该方法使用地址栈对栈指令的操作进行抽象的解释。但是与数据栈保留实际数据元素不一样，地址栈保留操作栈空间的指令地址。而可执行体所有可能执行序列形成的无穷大地址栈集可简单地用地址栈图来表示，通过地址栈图能够检测隐式的系统调用。

2　相关工作

反汇编是提取可执行体有用信息的第一步。Lakhotia 和 Singh[2] 利用静态分析得出一般恶意程序的攻击步骤。但是，Linn[3] 等展示的迷惑技术破坏了这种反汇编过程，使静态分析难以实施。

Collberg[4] 等对迷惑技术进行了详细描述并分类。迷惑引擎使得程序能够自动生成迷惑程序。Mistfall 和 Burneye 是两种迷惑引擎，前一个为二进制代码提供了迷惑引擎库，后一个是 Linux 下的封装工具。

但是，Barak [5] 等的研究表明没有一点瑕疵的迷惑技术是不存在的。事实上，检测具有迷惑技术的恶意程序的研究工作已经起步。Christodrescu 和 Jha[6] 使用抽象模式检测可执行体中的恶意模式。Mohammed[7] 研究了一种反迷惑技术，即撤消迷惑转换技术，比如对声明重排，变量重命名，表达式变形等迷惑技术进行反向撤消。在最近的研究中，Christodrescu[8] 等提出了一种通过语义直接检测恶意程序变形体的方法。该方法需要创建代码段的模版，以便于搜索各种病毒变形体。而 Lakhotia[9] 等也提出一种使用最大∏模式的串式分析找出恶意程序变形体的方法。与 Christodrescu 不同的是，该方法不需要使用模版。

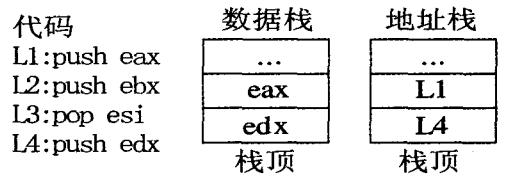

图 1　数据栈与地址栈

3　地址栈和地址栈图

地址栈是数据栈的抽象，数据栈保存后进先出(LIFO)的数据值，而地址栈保存 push 和 pop 等栈空间指令后进先出(LIFO)的地址序列。图 1 中，L1 到 L4 表示代码指令对应的

*)华为基金项目"基于安全行为模型的智能防御"。李毅超　硕士，副教授，主要研究方向：计算机网络及网络安全；何子昂　硕士研究生，主要研究方向：计算机网络及网络安全；曹　跃　硕士研究生，主要研究方向：计算机网络及网络安全。

地址，在执行完地址 L4 对应的指令后，数据栈和地址栈显示如图。首先，L1 和 L2 被依次压入地址栈，但由于 L3 的出栈指令，地址 L2 被弹出而地址 L4 被压入栈。

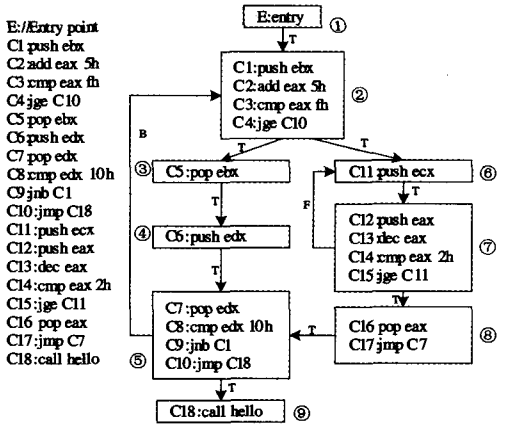

图 2　代码及控制路程图

地址栈和地址栈图通过程序控制流程图(CFG)获得。图 2 中显示了一段代码以及相应的控制流程图。在 CFG 中，一个分块代表一个程序点并且每块只包含一个栈操作指令以及其他控制转移指令，如：push，pop 或者 call 等。这里，将分块进行编号并且将每条控制边分别命令为 tree-edge，forward-edge，cross-edge 或者 back-edge。图 3 左半部分显示程序点 2 和程序点 4 的某段指令块序列所生成的地址栈。比如程序点 2 的第二个地址栈对应的指令块序列为：1→2→6→7→6→7，而程序点 4 的地址栈对应的指令块序列为：1→2→3→4→5→1→2→6→7。从图 3 左半部分中看出一段指令存在无数多个地址栈，那么对这么多地址栈进行跟踪是一项相当烦琐的工作。但是，一种更有效的方法为构造地址栈图。地址栈图是一个表示每个程序点所有地址栈集合的有向图。图 3 右半部分便是由图 2 中控制流程构造的地址栈图。

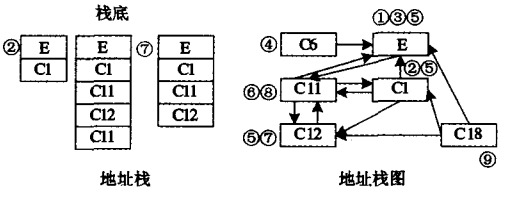

图 3　地址栈与地址栈图

4　构造地址栈图的算法

该算法是在程序控制流程图的基础上进行的，通过 IDA-Pro 等静态分析的商用软件都可得到程序控制流程图。

用控制流程图构造地址栈图的算法主要分为以下 7 步：

步骤 1：在 CFG 的每个分块中，用 push 和 pop 指令替代与之语义相同的栈空间操作指令序列。比如，如下指令：

```
des esp
des esp
mov [esp], eax
```

将用 push 指令替换。而 pop 指令将替换如下指令

```
mov eax, [esp]
inc esp
inc esp。
```

步骤 2：在 CFG 的每个分块中，检查是否有多于一个的栈空间操作指令(push，pop，call，ret)。如果有这样的分块，将这样的分块继续划分直到每个分块有且仅有一个栈空间操作指令。这样，每个分块在新的 CFG 中分别称作：push 块，pop 块，call 或者 ret 块。

步骤 3：对每个 CFG 中的边分别标识为：tree-edge，forward-edge，cross-edge 或者 back-edge。

步骤 4：遍历 CFG 并处理每个分块。

步骤 5：对于每个块 B，做如下处理：

5(a)：如果块 B 是 push 或者 call 块，就在地址栈图中创建一个结点，用 push(或 call)的指令地址标识该结点，并且将其编号。在 CFG 中，如果有一条 tree-edge 从另一块 B1 指向块 B，那么在地址栈图中创建一条边从 B 指向 B1(其中 B1 也是地址栈图中的一个结点)。重复这样的操作直到没有新的 tree-edge 边指向 B。

5(b)：如果块 B 是 pop 或者 ret 块，那么在 CFG 中找到每个通过 tree-edge 指向块 B 的块 B1。在地址栈图中所有 B1 指向的结点都被附加标识 B。

步骤 6：在 CFG 中，对于每个具有 forward-edge，cross-edge 和 back-edge 入点块也依次按照步骤 5 进行处理。

步骤 7：如果地址栈图被更新，重复步骤 1 到步骤 6 直到地址栈图不变化。

具体构造过程如下：

首先，在地址栈图中生成结点 E 并标识为①。然后在 CFG 图中找到下一块(push 块)，因此在地址栈图中生成块 C1、标识为②，并由于在 CFG 中有一条 tree-edge 从 E 指向 C1，那么在地址栈图中生成一条边从 C1 指向 E。同样地，push 块 C11 和 C12 也绘制到地址栈图中，并且相应的标识为⑥、⑦。下一块是 pop 块 C5，并且有一条输入的 tree-edge 来自块 C1，那么在地址栈图中块 C1 指向的块 E 被附加标识为③，因此 C6 指向标识为③的块 E。同样的，C11 被附加标识为⑧，C6 指向的块 E 也被附加标识为⑤。由于 C16 有 tree-edge 指向 C7，因此被标识为⑧的块 C11 在地址栈图中指向的块 C1 被附加标识为⑤。从而 C11 也指向被附加标识为⑤的块 E。接下来，考虑 forward-edge，cross-edge 和 back-edge。forward-edge 导致了 C11 指向 C12，因此⑧指向的块 C12 被附加标识为⑤。同时，被附加标识为⑤的块也指向 C11。所以，块 E 和块 C1 也指向 C11。back-edge 导致了 C1 指向被附加标识为⑤的块，因此 C1 指向 C12。而 C18 指向被附加标识为⑤的块 E、C1、C12。当有新的内容加入到 CFG 中时，将重复这个算法，将新信息更新到地址栈图中，直到重复的操作没有带来新的变化，得到一个稳定的地址栈图。

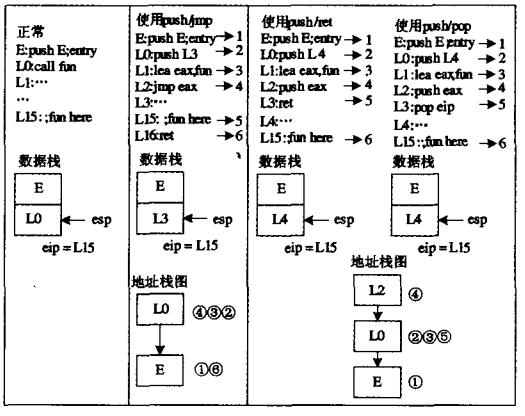

图 4　各种隐式 call 调用指令

同时，该算法是健壮的，能够应付 push 和 pop 循环中出现的栈失衡的情况。即在单个循环中 push 或者 pop 只发生一个，或者 push 和 pop 不平衡(push 指令比 pop 指令多，或者 pop 指令比 push 指令多)，这种算法仍能产生正确的地址栈图且包含所有程序点的地址栈。

5 隐式系统调用的检测

同段中的 call 调用是近过程调用。它将执行如下操作，如图 4 中正常 call 调用所示：将栈指针(esp)减去两个字节并将指令指针(eip)压入栈(eip，它包含了紧跟着 call 指令后的指令地址)，然后将要调用过程的地址插入到 eip 中。最后，一条 ret 指令从近过程调用中回退。

图 4 中，列出了三种隐式 call 指令调用技术。E，L0，L1 等存在于指令左边，它们代表指令的地址，而箭头"→"右边的数字代表控制流。在图 4 的中间列中，使用了 push/jmp 组合指令模拟 call 指令。指令 1 将入口点代码压栈，指令 2 将 call 指令后的指令地址压栈，即返回地址压栈，指令 3 装载函数 fun 的有效地址到寄存器 eax，指令 4 跳转到 fun 函数相应的地址。当 fun 函数返回时，控制权返回到指令 2 压入栈的返回地址处。因此，没有显示地使用 call 指令，但是达到了正常 call 指令调用的目的。通过该代码的地址栈得到的地址栈图可以检测这种隐式调用的迷惑手段。在地址栈图中，指令 6 回溯到指令 2 中最初压入栈的返回地址。通过对指令语义的观察，检测出 call 指令的调用。

在图 4 的右边列中，结合 push/ret 或 push/pop 指令隐式地模拟 call 指令。前四条指令和上例中语义一样。在指令 5 执行完 ret 或 pop eip 后，便将指令 4 中压入栈的地址便弹入到 eip 中，控制权便转移到 fun 函数中，实现了隐式 call 指令调用。如图所示的地址栈形成的地址栈图可用来检测这种隐式调用的迷惑手段。在地址栈图中，指令 5 中回溯到在指令 2 中压入的返回地址，通过观察 eax 得出结论：函数 fun 被调用。

6 实验

在 Windows 平台下，用 IDAPro 反汇编工具对两种恶意代码样本进行反汇编，然后使用地址栈图的相关方法对其进行检测，得出以下结论：

在 Win32. Evol 病毒中，使用了 push/pop 组合指令隐式执行 call 指令以到达系统调用的目的(以 LoadLibraryA 为例)。首先它将函数名字装载进一个寄存器中，然后把它压栈，最后弹出栈将控制权转移。以下为 Win32. Evol 示例代码：

```
lea eax, [ebp+var_14]
mov dword ptr [eax], 'daoL'
mov dword ptr [eax+4], 'rbiL'
mov dword ptr [eax+8], 'Ayra'
mov byte ptr [eax+0ch], 0
push eax
…
pop
```

该病毒使用的 push/pop 组合指令和图 4 右边列中一样以相同的方式隐式地模拟 call 指令。

而在 Win32. Giri 病毒中，却使用了图 4 中间列中的 push/jmp 组合指令来隐式执行系统调用。以下为 Win32. Giri 中隐式调用 GetTickCount 的示例代码：

```
mov dword ptr [eax], 'TteG'
mov dword ptr [eax+4], 'Ckci'
mov dword ptr [eax+8], 'tnuo'
mov dword ptr [eax+0ch], 0
push eax
lea ebx, var-func
jmp ebx
…
ret
```

因此，通过图 4 右边列和中间列的地址栈图便能检测出这类病毒的隐式的系统调用。

所以，地址栈图相关的检测方法可以有效地检测出恶意代码中使用组合指令执行的隐式系统调用。

结论 本文提出了一种检测具有迷惑技术的隐式系统调用的方法。该方法使用操作栈空间相关指令的地址，生成一种能够表示所有程序点处栈变化的地址栈图，图中的每一条路径都表示一个地址栈。检测结果表明，使用地址栈图能有效地检测恶意程序中具有迷惑技术的隐式系统调用。但是迷惑技术并不只局限于 call 指令，而且还包括参数比较的迷惑技术，返回指令的迷惑技术等，在下一步的工作中，将对此做进一步的研究。

参 考 文 献

1 Jordon M. Dealing with Metamorphism [EB/OL]. In: Virus Bulletin, 2002. 4～6
2 Lakhotia A, Singh P K. Challenges in Getting Formal with Viruses [EB/OL]. In: VirusBulletin, 2003. 14～18
3 Linn C, Debray S. Obfuscation of Executable Code to Improve Resistance to Static Disassemby [C]. In: The 10th ACM Conference on Computer and Communication Security 2003, Washington D. C, 2003
4 Collberg C, Thomborson C, Low D. A Taxonomy of Obfuscating Transformations l [R]: [Department of Computer Science, University of Aukland Technical Report 148]. July 1997
5 Barak B, Goldreich O, et al. On the (Im)possibility of Obfuscating Programs [C]. In: Advances in Cryptology (CRYPTO'01), 2001
6 Christodrescu M, Jha S. Static Analysis of Executables to Detect Malicious Patterns [C]. In: The 12th USENIX Security Symposium (Security '03), Washington D. C, 2003
7 Mohammed M. Zeroing in on Metamorphic Viruses [C]. In: The Center for Advanced Computer Studies. Lafayette, LA: University of Louisiana at Lafayette, 2003
8 Christodrescu M, Jha S, et al. Semantics Aware Malware Detection [C]. In: IEEE Symp. Security and Privacy, 2005
9 Lakhotia A, Karim M E, et al. Phylogeny Using Maximal pi-Patterns [C]. In: The 14th EICAR Conf, 2005

(上接第 83 页)

息交换系统，采用基于硬件分区的安全隔离技术和基于 IP 报文还原的内容深度处理技术，实现了多个外部网络接入一个内部网络时外网间的安全隔离和内外网间的安全通信。系统的设计方案和关键技术对其它网络安全隔离系统的开发和应用也具有重要的参考价值。

参 考 文 献

1 Bajcsy R, Kesidis G, Levitt K. Cyber defense technology networking and evaluation [J]. Communications of the ACM, 2004, 47 (3): 58～61
2 Algirdas A, Jean-Claude L, Brian R, et al. Basic Concepts and Taxonomy of Dependable and Secure Computing [J]. IEEE Transactions on Dependable and Secure Computing, 2004, 1(1): 11～33
3 Matthew V, Philip K C. Learning nonstationary models of normal network traffic for detecting novel attacks [A]. Proceedings of KDD'02 [C], Edmonton, Alberta, Canada, 2002. 376～385
4 李贵山，戚德虎. PCI 局部总线开发指南[M]. 西安：西安电子科技大学出版社，2001
5 陈颖. 基于包过滤的个人防火墙的研究与实现[D]. 西安：西安交通大学研究生院，2002
6 田新广. 基于主机的入侵检测方法研究[D]. 长沙：国防科技大学研究生院，2005
7 孙宏伟，田新广，张尔扬. 一种改进的 IDS 异常检测模型[J]. 计算机学报，2003，26(11)：1450～1455
8 田新广，高立志，张尔扬. 新的基于机器学习的入侵检测方法[J]. 通信学报，2006，27(6)：108～114