动态指令流差分分析在恶意软件分析中的应用*

孙　明，谷大武，李卷孺，罗宇皓
（上海交通大学 计算机科学与工程系，上海 200240）

摘　要： 针对静态分析方法已不能满足安全分析的需求，而传统的动态分析技术不能快速定位关键信息，且分析效率不高，提出了一种动态指令流差分分析技术，描述了差分分析模型和分析方法。该分析技术能够高速有效地分析恶意软件的关键数据，识别加密算法，分析混淆代码的功能模块和数据扩散情况。通过实验对其可行性和高效性进行了验证。

关键词： 恶意软件分析；动态指令流；差分分析；数据流
中图分类号： TP309.2　　**文献标志码：** A　　**文章编号：** 1001-3695(2012)02-0658-03
doi:10.3969/j.issn.1001-3695.2012.02.068

Differential analysis on dynamic binary and its application in malicious code analysis

SUN Ming, GU Da-wu, LI Juan-ru, LUO Yu-hao
(*Dept. of Computer Science & Engineering, Shanghai Jiaotong University, Shanghai* 200240, *China*)

Abstract: Static binary analysis methods cannot meet the demand for malicious code analysis, and the traditional dynamic analysis approaches cannot effectively find the critical information among the huge amount of dynamic binary code. This paper gave a kind of differential analysis approach on dynamic binary code and provided its model and method. This approach could effectively extract the sensitive information from malicious code and make the function module or data spread understood. Finally, it provided an experiment based on differential binary analysis system, which validated the capability and efficiency of the approach.

Key words: malware analysis; dynamic code; differential analysis; dataflow analysis

随着网络应用的日益广泛，对网络上传播的应用程序进行安全分析已成为软件安全领域中最为广泛的需求之一。恶意程序在与传统的安全分析技术（杀毒、主动防御、防火墙）的对抗中不断进化，对自身的保护能力越来越强，传统基于代码静态分析的工具与方法已经不能满足安全分析的需求，这就要求软件安全分析技术也必须不断发展。近年来，国际国内软件安全分析研究均提出了大量基于动态分析技术的分析方法，如 Taint 分析技术[1]、程序特征提取技术[2]、运行时程序切片技术等。这些技术在恶意软件分析、软件关键信息提取和运行时优化等方面都得到了广泛的应用。然而，上述动态分析技术依然存在很多问题，如动态信息量庞大难以快速定位关键代码和数据、代码混淆与变换导致程序特征信息不明显等。另外，如何将代码的动态分析和动态输入输出的数据结合起来也是一个重要的问题。尽管 Taint 分析可以部分解决这一问题，但是这种分析技术是从一个信息变量出发，分支不断扩展，代码分析量不断增加。而目前尚未出现一种更为精细的分析方法，能根据确定的数据有效地缩小代码的分析量。

基于软件动态指令流分析思想，本文提出了一种动态指令流差分分析技术，新技术通过匹配多条动态指令流轨迹之间的差异，分析软件运行时数据流扩散和指令对数据流的影响。与现有动态分析方法相比，该技术在分析二进制代码的密码算法、通信协议和数据扩散等方面有很大优势，明显减少了重点分析的指令流规模，能够快速定位软件中的某些关键信息，如密钥和协议结构等。常见的软件混淆技术是通过不同代码等价转换等方法对原指令转换，以增加代码被逆向分析的难度。但是动态指令流的差分受到这种转换的干扰程度较小，或者不受干扰，从而能够有效分析混淆代码。

本文的主要贡献包括：a）基于对差分分析思想理解，定义了动态指令流差分分析概念，即对于输入数据上存在差异的两次软件执行过程，分别记录其动态执行时的动态指令流和关联数据，通过分析输入数据的差异对软件执行时动态指令流的差异的影响，分析软件的关键数据位置和数据扩散情况，同时建立了动态指令流差分分析模型，介绍了指令差分和关联数据差分两种差分分析类型及其应用场景；b）提出了动态指令流差分分析方法，描述了对软件进行指令流差分分析的流程和技术细节，该分析方法通过动态指令流轨迹间的差异来定位数据和指令，其重点关注软件多次执行产生指令流的差异，以减少动态代码的分析量，同时软件的混淆、变换和多态技术对该分析技术的影响很小；c）实现了动态指令流差分分析原型工具，通过实验分析基于 OpenSSL 和 Crypto++ 库中 AES 算法模块的文件加密软件典型实现，验证了该分析方法能精准定位隐匿关键信息（密钥信息），提高分析的自动化程度，减少代码分析量，同时反映了软件执行过程中输入数据扩散情况。

1　相关背景

1.1　动态指令流分析技术

动态流指令分析技术是指通过虚拟机或模拟器对软件执

收稿日期：2011-07-09；修回日期：2011-08-11　　基金项目：国家自然科学基金资助项目（61073150）；SafeNet 东北亚科研计划资助项目
作者简介：孙明（1986-），男，硕士研究生，主要研究方向为通信安全与软件安全（iespresso@126.com）；谷大武（1970-），男，教授，博导，主要研究方向为密码学与计算机安全；李卷孺（1985-），男，博士研究生，主要研究方向为密码学、软件与网络安全.

行过程进行仿真,仿真环境对于上层运行的应用程序透明,任何基于软件的检测技术从理论上都不能区分。实际应用中,其工作基于 Bochs[5] 动态指令采集分析系统[4](简称 Bochs 系统),获取软件运行时的指令、内存、寄存器和 I/O 等信息,能够充分保证获得的动态指令流的原始性和关联数据的完整性。

文献[5,6]的设计和实现基于 QEMU 的动态分析系统,QEMU 运行速度较快,能够直接翻译部分二进制指令。Bochs 与 QEMU 的不同之处在于,Bochs 对于所有的 CPU 指令进行仿真,能更加真实地反映软件执行时 CPU 的状态和保持动态指令流的原始性。

1.2　软件关键信息提取

为了分析恶意软件的行为和实现机制,软件分析者需要挖掘隐藏在软件中的秘密关键信息。对于恶意软件中使用的密码算法,需要找到相关的密钥;对于恶意软件中使用的字符串如域名、IP 地址等信息,需要找到这些隐藏的关键字符串位置;对于通信交互协议,需要找到协议的结构和数据构造方式。为了提高恶意代码的生存能力,有效提高安全分析的复杂度和难度,隐藏关键代码和敏感数据,恶意软件作者通常采用反逆向分析技术对其进行保护,常用的保护方法主要是代码加密和代码混淆。

对于采用这些保护机制的代码,使用传统的静态分析已经无法有效获得程序中的秘密信息。现有的动态指令流分析对于软件中秘密信息的提取方式局限于特征识别,基于指令特征匹配的动态指令流分析技术[2,3],在动态指令流中匹配立即数或匹配字符串,定位关键数据的位置和数据扩散的影响位置。如果关键功能模块的特征不明显,或者软件开发者变换代码以淡化特征时,此技术识别的错误率很高,难以作出有效的判断。动态指令流差分分析方法通过分析动态指令流轨迹间的差异来定位数据和分析软件,其关注软件多次执行的差异,软件的混淆、变换和多态技术对该分析技术的影响很小。

2　动态指令流差分分析技术

2.1　动态指令流差分分析模型

动态指令流是指令和关联数据组成的序列,指令和关联数据可以充分反映软件执行过程中数据信息和控制信息的改变。动态指令流轨迹定义为序列 $S=\{\langle L_1,I_1,P_1\rangle,\langle L_2,I_2,P_2\rangle,\cdots\}$。其中,$L_i$ 表示该条指令在指令流序列中的偏移位置,I_i 代表该指令本身的信息,P_i 代表该指令所关联的寄存器或内存单元的值。动态指令流的差分是对序列 S 和 S' 的相同偏移位置 L_i 处的指令三元组进行差分操作,通常差分操作选择位异或运算。动态指令流差分可以分为指令差分和关联数据差分两种类型。

a) 指令差分。单条指令的指令差分可以用四元组 $\langle L,M,N,X\rangle$ 表示,如表 1 所示。其中: L 代表此指令在指令流轨迹的偏移位置; M 和 N 代表相对应的一组指令; X 表示指令中立即数的差分值,其颗粒度根据分析内容的不同选择字节或者双字。若动态指令流轨迹中相同偏移处指令 M 和 N 的差分值与预期差分值相匹配,将此四元组记录在差分序列(简称 Diff 序列)。Diff 序列包含所有因输入数据差异而产生差异的指令,可以分析输入的差异如何影响动态指令流,或者使用不同的常数,或者跳转到不同的函数。指令差分的特点在于能够快速定位受输入数据影响的跳转控制指令偏移处,帮助理解软件执行中所使用的不同立即数对于软件数据流产生的影响。当 Diff 序列出现无意义差分值时,说明两条轨迹此处偏移之前的控制流存在差异,从而发现输入数据对软件控制流的影响,可作为分析软件内部工作原理和分支跳转流程的辅助手段。

表 1　指令差分四元组示例

偏移 L	指令 M	指令 N	差分值 X
[0151]	PUSH 0xA4	PUSH 0xD4	0x00000070
[3041]	CALL 00042A50	CALL 00042B32	0x00000162
[7103]	ADD ESP, 0x12	ADD ESP, 0x14	0x06
[8375]	ADD ESP, 0x03	PUSH EAX	无意义

b) 关联数据差分。一个单条指令的关联数据差分可以用五元组 $\langle L,I,P,Q,X\rangle$ 表示,如表 2 所示。其中: L 表示此指令的指令流轨迹的偏移位置; I 代表指令本身; P 和 Q 代表指令 I 所关联的寄存器或内存单元的数据; X 表示 P 和 Q 的差分值,其颗粒度通常选择字节或双字。当 X 与预期的输入差分值相匹配时,认为该条指令对数据流间接产生影响,将此五元组记录到 Diff 序列。软件执行过程中的所有状态转移过程和数据信息都可以在动态指令流中体现[5],但由于软件动态运行时产生的动态指令流和关联数据的数据量庞大,很难定位为关键信息和发现影响数据流的指令集合。基于特征匹配(通常为指令的字符特征匹配或立即数匹配)来查找动态指令流中某些指令或者数据的方法,在软件特征受到混淆或者特征不明显时,成功率往往很低[4]。关联数据差分方式,优势在于能够快速过滤无关数据流的指令序列,减少需要分析的指令流的数据量,找到影响数据流的指令序列,从而定位软件中关键数据和分析数据扩散情况。

表 2　关联数据差分五元组示例

偏移 L	指令 I	关联数据 P	关联数据 Q	差分值 X
[1456]	PUSH EAX	EAX [000000EF] ESP [0012FD48]	EAX [000000EE] ESP [0012FD48]	EAX [00000001] ESP [00000000]
[4803]	XOR EDX, EBX	EDX [01F21464] EBX [64209CA0]	EDX [56381A00] EBX [64209CA0]	EDX [57CA0E64] EBX [00000000]

2.2　动态指令流差分分析方法

通过动态指令流定位和分析软件的关键数据和数据扩散,首先应获得原始准确的软件运行时指令流轨迹。本文应用 Bochs 模拟器的指令采集分析系统,仿真 CPU 执行过程,根据定义软件分析的重要模块或功能进行指令采集,按照线性地址、运行时间等条件过滤软件执行中无关的操作系统指令信息。在采集的指令流信息中定位需要分析的功能模块的偏移位置,用差分分析技术对此功能模块的指令流轨迹进行分析,最后对得到的结果进行验证。动态指令流差分分析流程如图 1 所示。

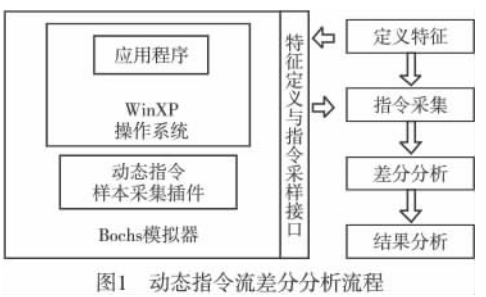

图1　动态指令流差分分析流程

a) 定义关键信息和重要指令类型。根据软件分析目的不同,设定动态指令流分析重点关注的指令类型或指令特征。通

常软件关键功能模块决定指令特征。例如分析密码算法,应该关注XOR和位运算指令的关联数据差分值;分析内存数据扩散和协议,应当选择所有读写内存单元指令的关联数据差分值;分析系统调用,可以关注PUSH、CALL、JMP等指令的差分值;分析程序状态转移,应选择CMP、JMP、CALL等指令的差分值。

b) 采集动态指令流轨迹。动态指令流轨迹包括动态指令序列和关联寄存器或内存单元数据序列。通过Bochs系统采集的动态指令信息包括指令的线性地址、指令、指令关联的寄存器和内存单元的值。动态指令流采集方式分为在线方式和离线方式,为了提高指令流采集的效率,采取在线模式匹配特征的方式过滤无关指令。Bochs模拟器加载相同的操作系统镜像文件,选择不同的输入数据作为软件的输入,对软件执行时动态指令采样。为提高指令数据的可读性,将二进制指令流轨迹反汇编成可读的动态指令流,同时将关联的寄存器和内存单元信息进行格式化。

c) 定位关键模块,进行差分分析。现代软件功能结构日趋复杂,捕获的动态指令流轨迹中,很难直接发现关键的功能模块的偏移位置。CFG和BFG[5,7]等技术可以用来划分指令流轨迹中的功能模块,找到其中调用关系,从而分析出各功能模块的行为。利用CFG技术和文献[6,7]中提出的若干分析方法,可以提取需要分析的关键功能模块的动态指令流轨迹偏移位置。此关键功能模块的动态指令流轨迹包含其所有动态信息,但实际分析实验中,其指令流数据量仍然很大,无法直接找到对数据流产生影响的指令和追踪数据扩散的影响。上述技术获得的关键模块动态指令流轨迹可能存在少许偏移误差,为了对其进行差分分析,必须先对若干软件关键模块的动态指令流轨迹进行同步和对齐,然后选取一对指令流轨迹进行指令差分和关联数据差分操作,同时将产生的差分值与预期的差分值进行匹配,将符合条件的指令信息及其偏移位置记录在Diff序列中。Diff序列中记录的指令和关联数据序列会对软件的数据流产生直接影响,可以定位软件中的对数据流产生影响的一些常量或者关键数据在动态指令流轨迹中的偏移位置,根据偏移位置在原始动态指令流轨迹中找到相关的指令序列,或者根据关联数据的变化在动态指令流轨迹中追踪输入差分数据的数据扩散。

d) 结果分析与验证。通过选取不同输入数据的多条动态指令流轨迹进行差分操作,或将获得的关键数据进行验证,保证分析结果的正确性。为了控制分析的效率和精度,必须合理划分分析粒度,通常选择以字节为单位进行差分分析,以符合CPU的处理粒度。分析粒度减小可能会增加分析的错误率,可以通过多次实验不同的差分输入来提高分析的准确度。

3 实验与分析

通常情况下,恶意软件会调用一些标准的密码学算法库进行加密运算,因此,本文选择常用的OpenSSL和Crypto++库AES模块进行分析。两者的AES模块都是比较典型的实现方法,前者采用硬编码S盒数据,攻击者可以直接在动态指令流的立即数中匹配S盒的特征,定位密码算法与S盒计算相关部分的偏移位置;后者代码采用动态方式计算生成S盒数据,同时在源码中使用大量内嵌汇编代码,使得其代码在编译后的指令特征相对较弱,采用特征匹配的分析方式将非常困难。

3.1 实验过程

分析基于上述AES模块的典型加密软件实现。首先定义关键功能模块为AES算法加密模块,关键信息是密钥,重要的指令类型为XOR、MOV等,差分操作为位异或运算。选择加密软件的输入数据P_1和P_2,其中P_1和P_2的某对应字节的差分值$X = P_1 \oplus P_2$。Bochs系统加载操作系统镜像和加密软件所在的磁盘镜像文件,操作系统启动后分别以P_1和P_2为输入运行加密软件,Bochs系统将记录该软件执行时的动态指令流轨迹,同时记录模拟CPU相关寄存器和相关模拟内存单元的值。在加密软件运行完成后,利用CFG和BFG等技术获得AES加密模块在动态指令流轨迹中的偏移位置,将该功能模块的轨迹提取并保存。为了清晰理解指令的功能特征,需要将二进制指令流轨迹进行反汇编,同时格式化相关数据信息。

将得到的两条AES加密模块的动态指令流轨迹进行指令差分和关联数据差分操作,同时将产生的差分值与输入数据的差分值进行匹配,将符合条件的指令信息及其偏移位置记录在Diff序列中。如表3所示,其中预期的字节差分值为0x01,Diff序列中指令及关联数据体现了差分的扩散。

表3 关联数据差分操作生成的Diff序列部分内容

偏移L	指令I	关联数据P	关联数据Q
[84142]	MOV BL, [EAX+F]	MR [0012FD5F] [000000EF]	MR [0012FD5F] [000000EF]
[84145]	OR EDX,EBX	EBX [000000EF]	EBX [000000EF]
[84146]	MOV EBX,EBP	EBX [000000EF]	EBX [000000EF]
[84147]	XOR EDX,EAX	EDX [2021E3EF]	EDX [2021E3EF]
[84149]	MOV EDI,EDX	EDX [5638DADF]	EDX [5638DADF]
[84163]	AND EBX,FF	EBX [5638DADF]	EBX [5638DADF]

软件分析的目的是获取AES密码算法的密钥,需要找到特征数据流和关联数据中与密钥相关的指令和数据。通过对AES算法本身的分析,在加密阶段有白化过程,即用原始密钥与明文进行异或操作,定位白化阶段的指令和关联数据,就可以直接读取密钥数据。在Diff序列中重点关注XOR指令,两条指令流轨迹第一次出现差异的位置可能为白化阶段,即表3中的偏移位置84147。最后如表4所示,在完整动态指令流轨迹84147偏移附近,重点关注XOR指令即可发现该加密软件所使用的密钥。同时表3中数据也反映了输入差异的扩散情况,即输入数据在指令流中的扩散轨迹,可以用来分析软件输入数据对于软件的执行过程的影响和分析软件的数据处理算法。

表4 定位指令流轨迹中含有密钥信息的指令偏移

偏移L	指令I	关联数据P	
[84106]	XOR EDX, EBX	EDX [F5BF8B37]	EBX [2627F685]
[84122]	XOR EBP, EDX	EDX [136F2E1F]	EBP [9715AD1D]
[84133]	XOR EDX, EDI	EDX [6BEC6F57]	EDI [D294DDC4]
[84147]	XOR EDX, EAX	EAX [76193931]	EDX [2021E3EF]

3.2 结果分析

实验表明,动态指令流差分分析技术能有效分析上述密码算法库的AES算法功能模块,明显降低分析的代码量,快速定位对影响数据流的指令,特别是在指令特征不明显时,差分分析仍然能够有效定位关键信息的位置,减少代码的分析数量。由于在明文中选取字节差分值作为识别特征,(下转第663页)

非线性几何攻击水印算法; 文献 [11] 的算法是一种使用傅里叶变换系数的抗局部非线性几何攻击水印算法; 本文是一种空域抗局部非线性几何攻击水印算法。相比而言，本文算法在大幅度降低算法运算量的同时，达到了较好的实验效果，同时本文算法提取的特征序列长度(64 bit) 低于文献 [10] 的算法(90 bit) 和文献 [11] 的算法(128 bit) 。

3.3　安全性分析

本文设计算法的安全性取决于用户密钥 k。用户密钥 k 和提取水印之间的依赖关系如图 3 所示。图 3 中横坐标为随机生成的 1 000 组独立的二值伪随机序列，纵坐标为 NC 值，为方便比较设置第 500 组为原始用户密钥。分别使用伪随机序列和用户密钥 k 进行水印提取，判决阈值 $T = 0.8$，可见本文算法对用户密钥是敏感的。

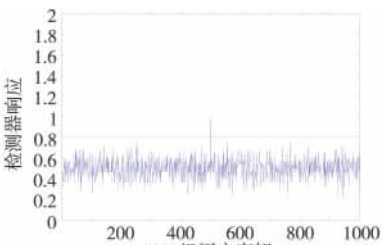

图3　检测可靠性测试结果

4　结束语

抵抗全局几何攻击的数字水印算法已经引起了信息安全领域科研工作者的广泛兴趣，至今为止已取得了一些重要的研究成果。但对于局部非线性几何攻击来说，相关的研究和成果都较少。本文设计了一种安全的基于空域图像轮廓形殊点抗局部非线性几何攻击的数字水印算法，即一种基于图像内容的安全鲁棒的抗局部非线性几何攻击数字水印算法。与当前的研究结果相比，本文算法可在大幅度降低算法复杂度的同时，取得较好的抗局部非线性几何攻击的性能。下一步的工作将集中于利用形殊点设计抗局部和全局几何攻击的水印算法。

参考文献:

[1] ZHENG Dong，LIU Yan，ZHAO Ji-ying. A survey of RST invariant image watermarking algorithms [J]. ACM Computing Surveys，2007，39(2)：1-91.

[2] FEI Chu-hong，KUNDUR D，KWONG R H. Analysis and design of secure watermark-based authentication systems [J]. IEEE Trans on Information Forensics and Security，2006，1(1)：43-55.

[3] ALTUN O，SHARMA G，CELIK U，*et al*. A set theoretic framework for watermarking and its application to semifragile tamper detection [J]. IEEE Trans on Information Forensics and Security，2006，1(4)：479-492.

[4] JIN Hong-xia，LOTSPIECH H. Hybrid traitor tracing [C] //Proc of IEEE International Conference on Multimedia and Expa. 2006：1329-1332.

[5] TIRKEL A Z，RANKIN G A，Van SCHYNDEL R M，*et al*. Electronic watermark [J]. Digital Image Computing Technology and Applications，Macquarie University，1993(1)：666-673.

[6] COX I J，KILIAN J，LEIGHTON F T，*et al*. Secure spread spectrum watermarking for multimedia [J]. IEEE Trans on Image Processing，1977，6(12)：1673-1687.

[7] VOLOSHYNOVSKIY S，DEGUILLAUME F，PUN T. Multibit watermarking robust against local nonlinear geometrical distortions [C] //Proc of International Conference on Image Processing. 2001：999-1002.

[8] 金聪，叶俊民，许凯华，等. 具有抗几何攻击能力的盲数字图像水印算法 [J]. 计算机学报，2007，30(3)：474-482.

[9] MITICHE A，AGGARWAL J K. Contour registration by shape specific points for shape matching [J]. Computer Vision，Graphics，and Image Processing，1983，22(3)：396-408.

[10] 崔得龙，左敬龙，彭志平. 基于范重心抗局部非线性几何攻击水印算法 [J]. 计算机应用，2010，30(8)：2161-2163.

[11] 李京兵. 基于变换域抗几何攻击数字水印算法研究 [D]. 重庆：重庆大学，2007.

(上接第 660 页) 可能会带来很小的错误概率，通过验证得到的密钥值或者使用不同软件输入数据分析，可以保证软件分析的准确性。

4　结束语

本文提出了动态指令流差分分析方法，详细描述了差分分析模型和分析方法。其优点包括利用动态指令流的差异明显减少代码的分析量、对于数据密集型恶意软件的分析优势明显、可以快速准确地剖析软件数据流和追踪数据扩散轨迹等方面。实验表明，采用动态指令流差分分析方法可以精准定位隐匿关键信息，同时提高分析的自动化程度，有效减少代码分析量，提高了分析效率。动态指令流差分方法依赖于输入数据的差分对程序跳转分支的影响，对于分支跳转指令密集型的软件分析优势不大，同时本文方法分析的粒度和差分匹配方式有待进一步研究和完善，这也是下一步的研究方向。

参考文献:

[1] NEWSOME J，SONG D. Dynamic taint analysis for automatic detection，analysis，and signature generation of exploits on commodity software [C] //Proc of Network and Distributed System Security Symposium. 2005.

[2] GRÖBERT F. Automatic identification of cryptographic primitives in software [EB/OL]. (2010) [2011-01-21]. http://kerckhoffs.googlecode.com/files/Groebert-Automatic.Identification of Cryptographic. Primitives. in. Software. pdf.

[3] Bochs: the open source IA-32 emulation project [EB/OL]. http://bochs.sourceforge.net/.

[4] 邓超国，谷大武，胡维奇. 一种基于动态指令流的恶意程序检测方法 [C] //全国计算机安全学术交流会论文集. 2010：173-179.

[5] SONG D，BRUMLEY D，YIN Heng，*et al*. BitBlaze：a new approach to computer security via binary analysis [C] //Proc of the 4th International Conference on Information Systems Security. Berlin：Springer，2008.

[6] CABALLERO J，YIN Heng，LIANG Zhen-kai，*et al*. Polyglot：automatic extraction of protocol message format using dynamic binary analysis [C] //Proc of the 14th ACM Conference on Computer and Communication Security. New York：ACM，2007：317-329.

[7] ALIOTO M，POLI M，ROCCHI S. A general model for differential power analysis attacks to static logic circuits [C] //Proc of IEEE Laternatioal Symposium on Circuits and Systems. 2008：3346-3349.

[8] 李卷孺，谷大武，陆海宁. 一种精简二进制代码的程序理解方法 [J]. 计算机应用，2008，28(10)：2608-2612.

[9] HARRIS L，MILLER B. Practical analysis of stripped binary code [J]. ACM SIGARCH Computer Architecture News，2005，33(5)：63-68.

[10] GODEFROID P，LEVIN M Y，MOLNAR D A. Automated whitebox fuzz testing [C] //Proc of Network and Distributed Systems Security. 2008.

[11] YIP A，WANG X，ZELDOVICH N，*et al*. Improving application security with data flow assertions [C] //Proc of ACM Symposium on Operating Systems Principles. 2009：291-304.

[12] YARDIMCI E，FRANZ M. Mostly static program partitioning of binary executables [J]. ACM Trans on Programming Languages and Systems，2009，31(5)：1-46.