

路径分支混淆研究综述

耿普, 祝跃飞

(战略支援部队信息工程大学, 河南 郑州 450001)

摘要: 代码混淆是一种便捷、有效的软件保护方法, 能够较好地对抗以逆向分析为基础的 MATE 攻击, 随着以符号执行为基础的自动程序分析技术的发展, 出现了能够抵抗符号执行的新代码混淆方法——路径分支混淆。依据路径分支信息的构成, 以及分支信息在对抗符号执行分析上的差异, 对分支混淆技术进行了分类, 并给出了分支信息泄露与符号执行的联系; 按照分支混淆的分类, 对当前分支混淆的研究进展进行了介绍和总结, 分析了各类分支混淆的优缺点; 最后, 对分支混淆技术的发展进行了展望。

关键词: 逆向工程; 代码混淆; 符号执行; 路径分支混淆

中图分类号: TP311

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2020027

Review of path branch obfuscation

GENG Pu, ZHU Yuefei

Strategic Support Force Information Engineering University, Zhengzhou 450001, China

Abstract: Code obfuscation is a convenient and effective technique of software protection, which can combat MATE attacks based on reverse engineering. With the development of automatic programming analysis technology, a new code obfuscation technique which named path branch obfuscation was proposed to defeat symbolic execution. With the composition of branch information and how does the branch composition effect in execution technique, the branch obfuscation was classified into three types. Then, a detailed research and development situation of each branch obfuscation class was exhaustively described. At last, the prospects of branch obfuscation were discussed.

Key words: reverse engineering, code obfuscation, symbol execution, path branch obfuscation

1 引言

随着互联网和信息技术的发展, 网络信息技术已经渗入社会生产和生活的各个角落, 与社会生产和生活的结合日趋紧密, 对社会发展产生了较大影响, 计算机软件作为信息处理和人与人之间交

互的载体也在不断发展壮大。由于互联网架构的开放性和商业软件的分发性, 互联网安全和软件安全成为当前研究的热点和难点问题, 特别是在 MATE (man at the end) 攻击环境下, 攻击者完全掌控软件的运行环境和运行状态, 能够对运行环境和运行状态进行任意的查看和修改, 通过软

收稿日期: 2019-06-24; 修回日期: 2019-09-03

通信作者: 耿普, 23015636@qq.com

基金项目: 国家重点研发计划基金资助项目 (No.2016YFB0801601, No.2016YFB0801505)

Foundation Item: The National Key R&D Program of China (No.2016YFB0801601, No.2016YFB0801505)

论文引用格式: 耿普, 祝跃飞. 路径分支混淆研究综述[J]. 网络与信息安全学报, 2020, 6(2): 12-18.

GENG P, ZHU Y F. Review of path branch obfuscation[J]. Chinese Journal of Network and Information Security, 2020, 6(2): 12-18.

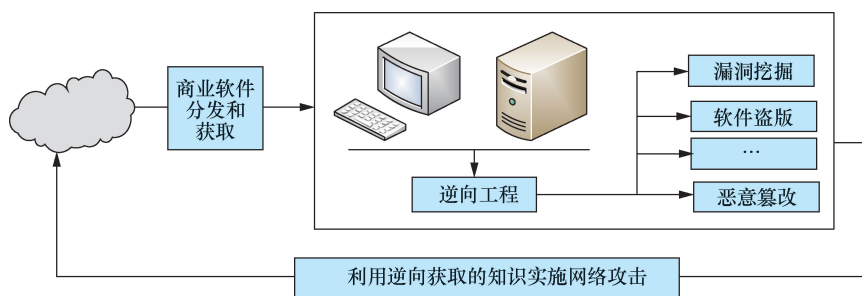


图1 MATE 攻击描述

Figure 1 The description of MATE attack

件逆向工程技术对软件进行分析、测试, 获取软件的执行逻辑、私密算法和数据, 利用软件漏洞挖掘及利用、软件盗版、软件篡改和软件私密算法及数据窃取等攻击手段攫取了巨大的非法利益, 给软件行业的发展带来了重大损失, 也给信息和网络安全带来了极大挑战。MATE 攻击描述如图1所示。

针对软件逆向工程和 MATE 攻击, 软件保护研究者提出了代码混淆^[1-4]的软件保护方法。通过对程序进行语义等价变换, 变换后的程序更加难以理解, 能更好地对抗逆向分析技术, 从而增加攻击者达成目标的时间和经济成本, 降低攻击收益, 迫使攻击者放弃攻击, 达成软件保护的目的。近年来, 随着计算和存储能力的爆发式增长, 以符号执行为基础, 结合污点分析、动态插桩、约束条件获取和约束条件求解等技术的软件自动分析方法得到了长足进步, 并且在漏洞挖掘与利用、软件测试、测试用例生成和代码重用等方面的应用取得了较好的效果。由于分支信息是程序执行逻辑和算法的基础, 而符号执行的前提是对分支信息进行分析, 获取约束条件进行求解, 因此安全研究者提出了一种防止分支信息泄露代码混淆技术——路径分支混淆。路径分支混淆对分支信息进行隐藏和保护, 实现程序执行逻辑的隐藏, 对抗符号执行^[5-18]和程序自动分析, 达成软件保护的目的。

2 符号执行与路径分支混淆

2.1 符号执行

符号执行是一种使用符号代替变量实际值进行代码执行分析的程序分析方法, 通过在条件跳转处, 使用符号值列出不同路径的约束条件, 最后对每条路径上的约束条件进行求解, 得到执行

该路径的输入符号对应的实际值, 为下一步的程序分析和测试提供良好的条件。

当前符号执行技术按照约束条件的获取方式, 可以分为传统符号执行、混合符号执行和选择符号执行。其中, 传统符号执行是在一次符号执行分析中, 得到所有路径的约束条件集。混合符号执行则在使用实际值执行的过程中对条件跳转指令和污染分析, 对当前执行路径上的所有分支列出对应的约束条件集。选择符号执行则是一种获取特定路径的输入符号值的程序分析方法。

2.2 代码混淆

Collberg^[1]等在对混淆变换的分类研究中, 对代码混淆给出了较为明确的定义, 并给出了代码混淆评价指标的描述。

定义1 代码混淆

对于任意的程序 P 和程序变换算法 T , 程序 P 经过算法 T 变换后得到的程序记为 P' , 若算法 T 满足下面的条件 1) 和条件 2), 则称算法 T 为代码混淆变换算法, 如图2所示。

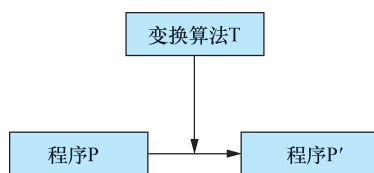


图2 代码混淆描述

Figure 2 The description of code obfuscation

1) 如果某输入导致程序 P 产生错误终止或者不能正常终止, 则相同的输入下程序 P' 可以终止或者不终止。

2) 若输入导致程序 P 正常终止, 则在相同的输入下程序 P' 必须正常终止且和程序 P 产生相同

的输出结果。

路径分支混淆是一种阻碍分支信息泄露的代码混淆技术，其通过阻碍分支信息泄露，抵抗符号执行。从符号执行的介绍可知，符号执行需要使用程序的分支信息来获取约束条件，并对约束条件进行求解得到对应路径的输入符号值。因此，通过隐藏条件跳转指令、复杂分支条件、阻断分支选择等方式，保护分支信息，从而阻碍符号执行对约束条件的获取和求解。路径分支混淆正是通过破坏约束条件获取和加大约束条件求解难度，从而对抗符号执行和隐藏程序执行逻辑。

3 路径分支混淆的分类

路径分支混淆通过对分支信息进行变换，在保持程序语义的同时，通过隐藏获取约束条件所需的分支信息、增加约束求解的难度甚至使约束不可解、切断符号执行的路径选择等方式，抵抗符号执行对程序进行分析。在结构上，分支信息包括分支控制指令、分支条件、分支目标地址 3 个部分，如图 3 所示。其中，分支控制指令用于对分支选择进行控制，根据分支条件的结果选择对应的分支代码执行；分支条件是分支选择的基础，根据分支条件计算的结果实现对分支选择的控制；分支目标地址是供分支选择的下一步执行地址，根据分支条件取值的不同，具有不同的分支目标地址。分支控制指令、分支条件和分支目标地址构成一个完整的分支信息。

按照分支信息的构成，把分支混淆分为 3 类，分别是分支控制方式混淆、分支条件混淆和分支

目标地址混淆。

分支控制方式混淆是一种对分支选择的控制方式进行混淆变换的代码混淆技术。在抵抗符号执行方面，该类混淆技术通过变换隐藏条件跳转指令，对抗符号执行的约束条件获取过程。由于当前的符号执行工具均是在条件跳转指令处，通过分析当前标志寄存器与输入符号之间的污染约束信息、跳转指令使用的标志寄存器位置信息，从而获取当前指令处不同分支选择的约束条件。因此，若对分支控制方式进行混淆变换，隐藏条件跳转指令，则当前大多数的符号执行工具无法列出正确的约束条件，从而使符号执行分析得不到正确的路径信息，对程序的分析得到错误结果。

分支条件混淆是当前分支研究的热点问题，其混淆变换的是分支信息中的分支条件。通过混淆变换，在保持程序语义的同时，程序执行时能够快速计算每个分支输入对应的分支条件取值，但通过分支条件取值难以逆向分析出对应的分支输入集合。分支条件混淆通过复杂化、单向化分支条件的计算过程，使当前的求解器无法完成对应路径的约束条件的求解，导致符号执行对程序的分析得不到结果，即分支条件混淆是一种对抗求解器求解能力的混淆方式。

分支目标地址混淆通过对分支目标地址的识别进行混淆变换，通过变换隐藏分支目标地址。符号执行在分支处通过对目标地址的定位，决定当前路径的下一条执行指令地址，若分支目标地址被隐藏，则符号执行由于不知道下一条指令地址而无法继续执行，从而中断符号执行的分析

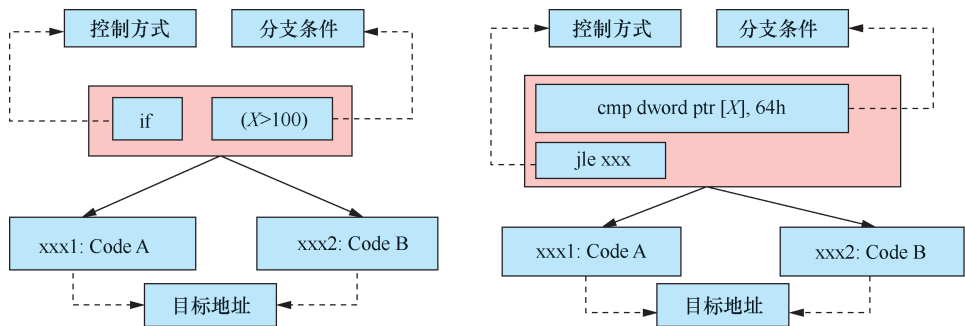


图 3 分支信息结构
Figure 3 The construction of branch information

过程。当然,在混合符号执行过程中,由于程序使用实际值执行,指令地址不可隐藏,且在分支点处获取约束条件用不到分支目标地址信息,因此分支目标地址混淆难以对抗混合符号执行。

4 路径分支混淆的研究进展

分支信息是程序执行逻辑的重要体现,也是程序控制权的转移依据,特别是随着以符号执行为基础的程序自动分析技术在逆向分析方面的发展,分支信息成为需要保护的重要信息,隐藏程序执行逻辑、阻碍分支信息泄露、对抗符号执行的分支混淆技术逐渐成为软件保护研究领域的热点问题。当前,在分支混淆研究领域已经取得一些较好的成果,在保护分支信息、对抗符号执行和隐藏程序执行逻辑等方面具有较好的软件保护效果。下面按照分支混淆技术的分类介绍其研究进展。

4.1 分支控制方式混淆的研究进展

分支控制信息是分支信息的重要组成部分,其决定着程序执行控制权的转移方式,当前最常见的控制转移方式是显式的条件跳转指令方式。程序源代码的控制逻辑经过编译器编译后大多使用条件跳转指令实现分支选择,如 `jz`、`jne`、`jb`、`jb` 等指令,这是一种显式的分支选择控制方式。通常,针对现有的符号执行工具通过显式的条件跳转指令分析,定位分支信息和获取约束条件,分支控制混淆通过异常处理、信号机制和指令副作用等多种方式实现控制权的隐式转移,避开显式的条件跳转指令,实现分支控制方式的混淆变换,使符号执行工具难以定位分支信息和获取约束条件。因此,分支控制方式混淆是一种显式控制到隐式控制的变换,分支控制方式的混淆变换如下。

1) 利用代码自修改方式实现条件跳转指令隐藏,保护分支信息,对抗反汇编技术和符号执行。使用垃圾数据对需要混淆的跳转指令进行覆盖,同时在该地址的前支配路径上添加自修改代码,使跳转指令在执行前被恢复,并在执行后重新使用垃圾数据覆盖跳转指令。垃圾数据不能反汇编的特点使程序具有对抗静态反汇编的功能;同时由于条件跳转指令被垃圾数据覆盖,静态代码与实际执行的代码不一致,导致使用静态代码

作为输入的静态分析技术得出错误结论,因此,该分支混淆方法能够对抗使用静态代码进行分析的传统符号执行和控制流图生成。Balachandran 等^[6]提出的代码混淆方法就是一种使用代码自修改实现的静态分支混淆技术。

2) 利用异常处理实现控制权转移,替代条件跳转指令,隐藏分支信息,对抗符号执行。当程序执行过程中发生异常时,操作系统会获取程序控制权进行异常处理,实现控制权在程序和系统之间的转移,代码混淆正是利用异常发生的控制权转移实现控制方式的混淆变换。分支混淆首先构造一段在特定条件下会产生异常的代码——条件异常代码,并使用条件异常代码替换跳转指令;通过注册异常处理函数,在控制权转移过程中把控制权转移到分支目标地址,实现分支混淆。由于条件异常代码是由常见的非跳转指令构成的,因此隐藏了条件跳转指令,使分支混淆能够隐藏执行逻辑和控制流、对抗传统符号执行。Popov^[19]、Lin^[7]和贾春福^[8]都使用异常处理分别提出了自己的分支混淆方法。Popov 在 Linux 系统下使用自陷指令替代无条件跳转指令产生信号,使预先注册的信号处理函数对控制权转移进行干涉,使分支目标地址能够正确获取执行控制权;使用不可反汇编的垃圾数据覆盖无条件跳转指令,对抗反汇编技术。Lin 和贾春福在 Popov 的基础上,使用常见的 `mov` 等普通指令构造条件异常代码,并替换条件跳转指令;通过异常处理函数处理控制权转移,实现分支混淆,对抗符号执行。

分支控制方式混淆使用隐式控制转移替代显式控制转移,阻碍符号执行工具对约束条件的获取,即只是增加了获取的难度,并非使获取约束条件成为不可能。因此,通过修改符号执行工具的分支信息定位方式和分支信息分析方式,可以获取被混淆分支的约束条件。因此,其后续的研究重点应该在如何提高混淆强度和增强混淆的隐蔽性上。只有通过增强混淆的隐蔽性,提高修改符号执行工具获取被混淆分支地址的难度,才能提高分支混淆对抗符号执行和逆向分析的强度。例如,使用异常处理替代条件跳转分支混淆,如果用于混淆的异常处理和正常使用的异常处理被区分开,即可以通过代码动态插桩的方式,在异

常发生处和异常返回程序空间地址处获取分支信息,实现混淆还原。

4.2 分支条件混淆的研究进展

分支条件混淆是一种对分支条件进行变换的代码混淆技术,当前的分支条件混淆主要是基于哈希和加密算法的分支输入验证法和基于机器学习的分支输入分类法。基于哈希函数和密码算法的分值输入验证法则是利用哈希函数和密码算法的单向特性和复杂性,阻碍分支信息泄露和对抗约束求解;基于机器学习的分值输入分类法利用机器学习结果的复杂性和不可理解性,实现程序执行逻辑隐藏和约束条件复杂化,对抗符号执行和逆向工程。分支条件的混淆变换如下。

1) 使用哈希函数和加密算法对分支条件进行语义等价的不可逆混淆变换,实现分支条件混淆和程序执行逻辑隐藏。哈希函数和加密算法都具有相当的复杂性,且哈希函数具有良好的单向特性。基于哈希函数和加密算法的混淆分支对应的约束条件求解难度等同于解密和通过哈希值求解哈希输入的难度,当前条件下,约束求解器难以完成此类约束条件的求解。因此,基于哈希函数和加密算法的分支混淆方法具有非常高的混淆强度和混淆弹性,且解密和哈希计算的速度相对较快,使分支混淆的消耗较为低廉,具备较高的执行效率。在该类分支混淆技术中,Sharif^[9]使用哈希值相等替代原等于条件实现分支条件混淆,并且使用分支输入作为密钥对条件代码进行加密,实现了能够对抗恶意分析的代码保护方法。Wang^[12]等使用保留前缀加密算法和哈希函数对分支的输入集合进行处理,通过加密前缀匹配替代分支条件,实现了分支条件混淆从等于分支到大小比较分支的扩展。

2) 基于机器学习训练分值输入分类器,使用分类器替代分支条件,利用分类器的复杂性和不可理解性隐藏程序执行逻辑,实现分支条件混淆。机器学习是当前研究的热点问题,通过使用多层神经网络、支持向量机等各种方法对数据分类进行训练,得到的分类器具有较优的分类效果。但分类器的运行逻辑却难以被理解,这在机器学习是一个缺点,但在对抗逆向分析中则是优点。

Zong^[13]、Ma^[14]和陈喆^[15]等分别基于支持向量机、神经网络和随机森林对分支输入进行分类训练,并使用分类器替代分支条件,实现分支混淆;Ma^[16]等进一步使用分支条件混淆实现了抗分析攻击的软件指纹嵌入,提升了软件追溯盗版复制的能力。

3) 基于代码分离的分支条件混淆。上述两类分支条件混淆是对分支条件的逻辑进行混淆,通过逻辑变换对抗约束求解。而基于代码分离的分支条件混淆则是一种物理隔离的代码保护方法,通过把分支条件的执行逻辑移动到远端可信实体,不可信实体则通过与可信实体的通信获取分支条件的执行结果。Wang等^[17]通过该方法实现了分支条件混淆,并结合分支控制方式混淆,使用垃圾数据覆盖条件跳转指令,对抗反汇编技术。

4.3 分支目标地址混淆的研究进展

分支目标地址是决定符号执行下一条指令地址的关键信息,通过分支目标地址隐藏可以对抗传统符号执行。由于混合符号执行使用实际值运行程序,单纯的目标地址混淆对混合符号执行不造成影响,因此分支目标地址混淆的相关研究较少。当前的目标地址混淆主要是通过代码移动方式实现的,如陈喆等^[18]把控制逻辑和分支目标地址移动至可信实体执行,实现分支条件和分支目标地址的混淆。

5 路径分支混淆的发展趋势研究

5.1 分支混淆技术的优缺点分析

分支控制方式混淆优点如下。① 混淆后程序具有非常高的执行效率,几乎与混淆前的执行效率相当。② 能较好地对抗反汇编技术。由于控制方式的变换导致跳转指令消失,容易通过增加垃圾数据对抗反汇编技术。③ 增加约束条件获取难度。缺点是:① 混淆弹性有待提高,难以抵抗程序动态分析技术;② 使用动态分析手段容易还原出被混淆的分支信息,实现混淆恢复。

分支条件混淆的优点如下。① 具有较好的混淆强度和混淆弹性。混淆本身基于变换的不可逆性和难以理解性,因此混淆后的分支条件难以被理解和还原。② 对抗符号执行的效果较好。混淆

针对的是求解器的求解能力,而非符号执行工具的运行方式,因此该混淆是一种治本的方法。

③ 抗逆向分析效果好。难以理解和单向性保证了分析者获取信息的难度。其缺点是执行效率相对较低。特别是基于机器学习的分类器方法,在执行时间和数据、代码空间的消耗上还需要进一步优化。

分支目标地址混淆的优点是:① 实现简单,能对抗传统符号执行;② 能对抗控制流图生成。其缺点是:① 难以抵抗混合符号执行;② 使用IO通信导致执行效率相对较低。

5.2 分支混淆发展展望

当前在分支混淆研究方面已经有了一些较好的成果,并且一些成果具有较好的实际应用效果,较好地提升了代码保护的功效,为软件、信息和网络安全做出了贡献。但当前对分支混淆的研究还不够充分和深入,需要更多的科研工作者和更多的科研精力投入。分支混淆主要解决分支信息泄露问题,阻碍符号执行技术对程序进行分析;同时增加逆向分析者对混淆后代码的理解难度,消耗攻击者在程序分析过程中消耗的资源。为更好地使用分支混淆技术实现代码保护,今后分支混淆在以下几个方面值得深入研究。

1) 基于不同类别的分支混淆具有不同的优缺点。因此,研究不同类别的分支混淆技术的融合,扬长避短,可以提高分支混淆的执行效率、混淆弹性和混淆强度。

2) 分支混淆仅仅是代码混淆技术中的一个分支,其保护对象仅限于分支信息,局限了代码保护的范围。因此,需要研究分支混淆和其他代码混淆技术的融合,提高混淆技术的保护范围。

3) 加强分支条件混淆的研究。分支控制方式混淆和分支目标地址混淆针对的是符号执行工具的运行方式,并没有从根本上解决分支信息泄露的问题。针对约束求解器能力,则从根本上解决约束不可解的问题。因此,分支条件混淆在隐藏执行逻辑和对抗符号执行上具有更高的混淆强度和混淆弹性,具备更好的研究价值。

因此,在下一步的研究中,可以针对分支条件混淆方向采取重点研究,提高执行效率和混淆强度;同时融合不同类别分支混淆技术的优点,以及结合其他混淆技术的优点,进行综合研究,更好地发挥代码混淆的优势,提高软件保护能力,为软件、信息和网络的安全提供保障。

6 结束语

分支混淆通过对代码中的分支信息进行保护,提升获取路径约束和约束求解的难度,从而使以符号执行技术为基础的程序分析技术失效,达成代码保护和对抗符号执行的目的。

本文综合分析了当前分支混淆技术的研究内容,针对分支混淆技术研究的侧重点和混淆目标的不同,对分支混淆进行了分类;按照本文提出的分支混淆分类,详细介绍了当前分支混淆研究的内容和进展情况,分析和总结了每类分支混淆技术的优缺点;最后,总结了分支混淆的研究内涵,展望了分支混淆技术的发展方向,为下一步的分支混淆技术研究提供参考。

参考文献:

- [1] COLLBERG C, THOMBORSON C, LOW D. A taxonomy of obfuscating transformations[R]. Department of Computer Science, The University of Auckland, New Zealand, 1997.
- [2] MAVROGIANNOPOULOS N, KISSERLI N, PRENEEL B. A taxonomy of self-modifying code for obfuscation[J]. Computers & Security, 2011, 30(8): 679-691.
- [3] AVIDAN E, FEITELSON D G. From obfuscation to comprehension[C]//2015 IEEE 23rd International Conference on Program Comprehension. 2015: 178-181.
- [4] 杨宇波. 代码混淆模型研究[D]. 北京: 北京邮电大学, 2015.
YANG Y B. Research on code obfuscation model[D]. Beijing: Beijing University of Posts and Telecommunications, 2015.
- [5] BANESCU S, COLLBERG C, GANESH V, et al. Code obfuscation against symbolic execution attacks[C]//The 32nd Annual Conference on Computer Security Applications. 2016: 189-200.
- [6] BALACHANDRAN V, EMMANUEL S. Potent and stealthy control flow obfuscation by stack based self-modifying code[J]. IEEE Transactions on Information Forensics and Security, 2013, 8(4): 669-681.
- [7] LIN H, ZHANG X, YONG M, et al. Branch obfuscation using binary code side effects[C]//International Conference on Computer, Networks and Communication Engineering (ICCNC 2013). 2013.

- [8] 贾春福, 王志, 刘昕, 等. 路径模糊: 一种有效抵抗符号执行的二进制混淆技术[J]. 计算机研究与发展, 2011, 48(11): 2111-2119.
- JIA C F, WANG Z, LIU X, et al. Branch obfuscation: an efficient binary code obfuscation to impede symbolic execution[J]. Journal of Computer Research and Development, 2011, 48(11): 2111-2119.
- [9] SHARIF M I, LANZI A, GIFFIN J T, et al. Impeding malware analysis using conditional code obfuscation[C]//NDSS. 2008.
- [10] 王志. 二进制代码路径混淆技术研究[D]. 天津: 南开大学, 2012.
- WANG Z. Research on binary code obfuscation[D]. Tianjin: Nankai University, 2012.
- [11] 王志, 贾春福, 刘伟杰, 等. 一种抵抗符号执行的路径分支混淆技术[J]. 电子学报, 2015, 43(5): 870-878.
- WANG Z, JIA C F, LIU W J, et al. Branch obfuscation to combat symbolic execution[J]. Acta Electronica Sinica, 2015, 43(5): 870-878.
- [12] WANG Z, MING J, JIA C F, et al. Linear obfuscation to combat symbolic execution[C]//European Symposium on Research in Computer Security. 2011.
- [13] ZONG N, JIA C F. Branch obfuscation using "black boxes"[C]//2014 Theoretical Aspects of Software Engineering Conference. 2014: 114-121.
- [14] MA H, MA X, LIU W, et al. Control flow obfuscation using neural network to fight concolic testing[C]//International Conference on Security & Privacy in Communication Systems. 2014.
- [15] 陈喆, 贾春福, 宗楠, 等. 随机森林在程序分支混淆中的应用[J]. 电子学报, 2018, 46(10): 156-164.
- CHEN Z, JIA C F, ZONG N, et al. Branch obfuscation using random forest[J]. Acta Electronica Sinica, 2018, 46(10): 156-164.
- [16] MA H, LI R, YU X, et al. Integrated Software fingerprinting via neural-network-based control flow obfuscation[J]. IEEE Transactions on Information Forensics & Security, 2016, 11(10): 2322-2337.
- [17] WANG Z, JIA C F, LIU M, et al. Branch obfuscation using code mobility and signal[C]//IEEE Computer Software & Applications Conference Workshops. 2012.
- [18] 陈喆, 王志, 王晓初, 等. 基于代码移动的二进制程序控制流混淆方法[J]. 计算机研究与发展, 2015, 52(8): 1902-1909.
- CHEN Z, WANG Z, WANG X C, et al. Using code mobility to obfuscate control flow in binary codes[J]. Journal of Computer Research and Development, 2015, 52(8): 1902-1909.
- [19] POPOV I V, DEBRAY S K, ANDREWS G R. Binary obfuscation using signals[C]//USENIX Security Symposium. 2007: 275-290.

[作者简介]



耿普(1982—), 男, 云南宣威人, 战略支援部队信息工程大学博士生, 主要研究方向为网络与信息安全、代码保护。



祝跃飞(1962—), 男, 浙江杭州人, 博士, 战略支援部队信息工程大学教授、博士生导师, 主要研究方向为网络空间、公钥密码。