

基于 gadget 特征分析的软件多样性评估方法

谢根琳, 程国振, 王亚文, 王庆丰

(信息工程大学, 河南 郑州 450001)

摘要: 软件多样性能够有效提升系统弹性和安全性, 广泛应用于软件分发、操作系统等场景, 但现有软件多样性评估方法普遍基于常规代码特征进行度量且较为单一, 难以准确反映软件多样性带来的安全增益。针对此问题, 以 ROP (return-oriented programming) 攻击视角出发, 通过分析软件多样性对代码重用攻击过程中各阶段的影响, 提出了一种综合 gadget 质量、实用性和分布指标的软件多样性评估方法, 通过度量软件多样性对构建 gadget 攻击链的难度、攻击者潜在可获得的计算能力和攻击者在不同变体中搜索 gadget 成本的影响程度对其进行安全性评估。利用不同粒度的多样化技术进行实验, 结果表明, 所提方法能够准确全面地反映软件多样性带来的安全增益, 细粒度的多样化技术能使软件中大量的 gadget 重定位/修改/移除, 增加攻击软件变体的成本, 但部分多样化技术会导致不同程度的软件膨胀等问题。最后, 依据不同度量指标下得到的结果对现有软件多样化技术的优势与不足进行分析和讨论。

关键词: 软件多样性; 代码重用攻击; gadget 特征; 安全增益

中图分类号: TP309

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2023047

Software diversity evaluating method based on gadget feature analysis

XIE Genlin, CHENG Guozhen, WANG Yawen, WANG Qingfeng

Information Engineering University, Zhengzhou 450001, China

Abstract: Software diversity is commonly utilized in scenarios such as software distribution and operating systems to improve system resilience and security. However, existing software diversity evaluation methods are typically based on conventional code features and are relatively limited in scope, which can make it difficult to accurately reflect the security benefits of software diversity. To address this issue, a software diversity evaluation method was proposed from the perspective of ROP attack by analyzing the impact of software diversity on the difficulty of building a gadget attack chain, the attacker's potentially available computing power, and the attacker's cost of

收稿日期: 2022-06-28; 修回日期: 2022-11-03

通信作者: 程国振, guozhencheng@hotmail.com

基金项目: 国家重点研发计划 (2021YFB1006200, 2021YFB1006201); 国家自然科学基金 (62072467)

Foundation Items: The National Key R&D Program of China (2021YFB1006200, 2021YFB1006201), The National Natural Science Foundation of China (62072467)

引用格式: 谢根琳, 程国振, 王亚文, 等. 基于 gadget 特征分析的软件多样性评估方法[J]. 网络与信息安全学报, 2023, 9(3): 161-173.

Citation Format: XIE G L, CHENG G Z, WANG Y W, et al. Evaluating software diversity based on gadget feature analysis[J]. Chinese Journal of Network and Information Security, 2023, 9(3): 161-173.

searching for gadgets in different variants. Metrics for the quality, practicability, and distribution of gadgets were integrated into this method. Testing was conducted using diversity technologies with different granularity. The evaluation results showed that the proposed method could accurately and comprehensively reflect the security gain brought by software diversity. It was observed that software diversity could relocate/modify/delete a large number of gadgets in the software, increasing the cost of attacking different software variants but also leading to different degrees of software expansion. Finally, an analysis and discussion of the advantages and disadvantages of existing diversity techniques were conducted based on the experimental results.

Keywords: software diversity, code reuse attack, gadget feature, security gain

0 引言

代码复用和自动更新等软件开发步骤为软件分发、bug 报告等软件的推广与优化提供了极大的便利,但导致“软件单一文化”问题的出现^[1],给如今庞大的软件市场带来了严重的风险,使攻击者针对某一软件的攻击经验可成功复用在其他运行相同软件的用户上。代码重用攻击是其中被广泛使用的一种攻击方法,即重新排列合法代码序列,并通过重定向程序的控制流至该代码片段处,实现破坏系统或窃取信息等目的。其中,用于执行恶意行为的合法代码序列段被定义为 gadget,每个 gadget 以控制流跳转终止,依据该跳转指令的差异有多种不同类型的代码重用攻击被提出。例如,面向返回编程(ROP, return-oriented programming)攻击^[2]以返回指令结束,面向调用编程(COP, call-oriented programming)攻击^[3]以调用指令结束,面向跳转编程(JOP, Jump-oriented Programming)攻击^[4]以跳转指令结束。

除上述攻击外,JIT-ROP(just-in-time ROP)攻击^[5]在程序运行时部署,不依赖于任何脱机静态分析,动态地发现 gadget,可绕过众多的防御技术,被广泛用于浏览器等应用进行漏洞利用,导致网络和软件系统中存在严重的安全隐患。

为有效应对“软件单一文化”问题和代码重用攻击带来的威胁,软件多样化技术通过在软件开发周期的不同阶段(开发、编译、加载和执行)部署软件代码变体^[6],增加不可预测性,提升软件系统的弹性和安全性^[7],实现对不同软件变体的攻击经验难复用、攻击效果难复现,进而显著提高攻击成本,降低大规模攻击风险。其中,地址空间布局随机化(ASLR, address space layout randomization)^[8]已被广泛应

用于操作系统,Linux 中通过内核参数 randomize_va_space 对不同级别的 ASLR 进行控制,在程序加载时,随机排列堆、栈等模块的基址。ASLR 的内存布局如图 1 所示。

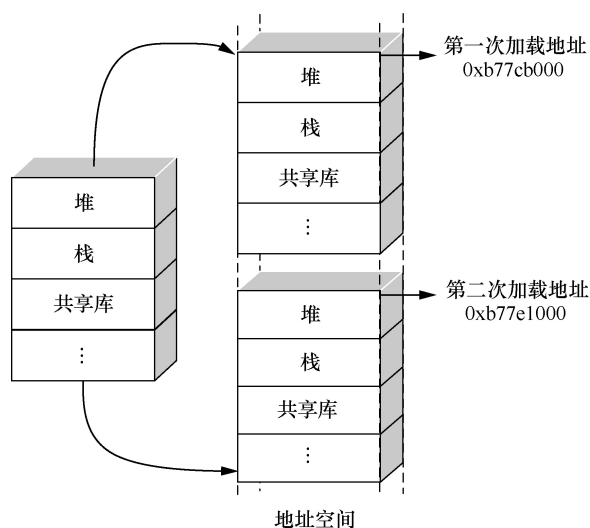


图1 ASLR 的内存布局
Figure 1 The memory layout of ASLR

尽管当前软件多样化技术发展迅速,其与内存权限、异常检测等防御技术相结合的多种机制被提出^[9-10],但对于如何具体全面地衡量软件多样化技术所带来的安全增益仍是一个难题。该研究具有十分重要的价值,可用于指导网络和软件系统中部署更具有安全性的多样化技术,防御 Web 浏览器等应用中日益增加的漏洞攻击^[11]。

软件多样性评估直观上为衡量其引入的变体间差异度,文献[12]通过计算多样化技术部署前后包含 TF-IDF 和 doc2vec^[13]等信息的软件特征向量的欧氏距离评估不同多样化技术的效果,但并未给出该指标与安全性间的相关度,不能直接反映软件多样性带来的安全增益。文献[14-15]计算变体间共享的 ROP 类 gadget 占比本质上是计算

变体指令的偏移量, 尽管该指标反映了软件多样性对攻击流程中搜索 gadget 步骤的影响, 与安全性强相关, 但并未全面考虑软件多样性对代码重用攻击构建攻击链等步骤的影响, 所以不能准确地全面地反映软件多样性带来的安全增益。

针对上述问题, 本文提出了一种基于 gadget 特征分析的软件多样性安全评估方法, 全面地考虑了软件多样性对代码重用攻击各阶段的影响。主要的研究工作及贡献如下。

1) 提出 gadget 质量、实用性和分布 3 个度量指标, 通过计算具体的质量评分、不同 microgadget 集合计算能力和基于核密度估计的 gadget 分布差异, 直接度量软件多样性带来的安全增益。

2) 依据本文提出的评估方法对现有不同粒度的软件多样化技术进行实验评估, 从多个角度展示了不同多样化技术对软件安全性的影响。

1 相关工作

现有的软件多样性评估方法从不同特征角度可以概括为 3 类: 相似性、弹性和防御攻击能力。

针对源码文件的软件多样性评估方法主要从相似性角度进行度量^[16], 依据切分单元等差异, 可分为基于图结构^[17]、串结构^[18]和度量值 (如 McCabe 圈复杂度等)^[19-20]。但是上述方法仅度量了软件多样性引入的变体间差异度, 并没有反映多样性带来的安全增益, 也没有计算所提相似性指标与安全性间的相关度。

从二进制层级对软件多样性进行评估是研究的主要方向, 涉及相似性、弹性和防御攻击能力等指标。

1) 相似性指标: 主要使用熵、哈希值等进行软件多样性评估。文献[21-22]对应用于软件多样性评估的熵进行了总结分类, 定义了全局熵、函数熵等。但是当存在指针泄露、元数据信息泄露时, 评估结果是不准确的, 往往会高估软件多样性提供的安全性。

2) 弹性指标: 软件多样化技术部署后会生成原始软件对应的软件变体, 而弹性指标量化通过反汇编等工具处理软件变体时所表现的随机性因素或错误, 进而评估软件多样化技术带来的安全增益。文献[23]使用 IDA Pro 6.2 对多样化技术处

理过的软件进行反汇编, 计算其指令的反汇编错误率和控制流错误率以进行软件多样性评估。文献[24]通过 KLEE 等工具计算多样化技术部署前后符号执行时间的差异, 从而评估软件多样性的效果。但是上述方法仅考虑软件多样性对逆向工程的影响, 并未体现软件多样性对代码重用等攻击的影响, 且符号执行覆盖全部分支受硬件性能等影响, 所需时间较长。

3) 防御攻击能力指标: 该指标的引入是由于多样化技术的主要用途是防御代码注入/重用等攻击。从防御攻击能力角度展开评估主要使用逻辑论证、具体攻击测试和特定攻击的代码分析等方法^[6]。

逻辑论证通过论证攻击所必须的信息或途径被防御机制隐藏或阻断, 则认为防御机制有效。文献[9]提出 Readactor 机制, 在对其有效性进行评估时, 从代码重用攻击的角度, 论证其对 5 种不同的代码重用攻击的防御效果。但是逻辑论证具有片面性, 因此通常将其与攻击测试相结合。

具体攻击测试通过构建多种类型的攻击, 测试部署软件多样化技术前后攻击的差异。文献[25]在网络安全比赛测试平台中部署多样化编译器, 利用其中不同的多样化技术, 对软件多样性进行评估。尽管通过具体攻击测试能够准确反映软件多样性提供的安全性, 但其开销大、门槛较高、普适性较差。

特定攻击的代码分析主要指根据代码重用攻击中 gadget 的相关性质定义度量标准。文献[15]定义 gadget 存活率 (多样化技术部署前后未产生偏移的 gadget 占比) 为评估指标。文献[26]分别计算 gadget 减少率、损坏率和收敛时间等, 评估重随机化技术。但上述方法均未能全面地反映软件多样性对整个代码重用攻击流程中各攻击步骤的影响, 具有一定的片面性。

2 评估模型

代码重用攻击通过在可执行文件或程序执行过程的内存中搜索所需 gadget 以构建攻击链, 本文提出的软件多样性评估方法提取受软件多样性影响的 gadget 特征, 设置相应的度量指标量化分析软件多样性带来的安全增益。本节对受影响的 gadget 特征、基于 gadget 度量指标和软件多样性

评估算法分别进行描述。

2.1 gadget 特征提取

代码重用类攻击的流程如图 2 所示, 其中, ROP/JOP/COP 攻击的部署需要对应用程序的二进制文件进行静态脱机分析以识别所需 gadget, 而 JIT-ROP 攻击则在程序运行时动态地遍历内存中的代码页以获得所需的 gadget。由图 2 可知, 软件多样性对所有代码重用类攻击的影响主要集中于内存泄露、gadget 搜索和构建 gadget 攻击链的步骤, 所以软件多样性对 ROP/JOP/COP 攻击和 JIT-ROP 攻击的影响可以通过提取相同的 gadget 特征进行度量。

1) gadget 搜索步骤: 软件多样性通过改变内存空间布局影响 gadget 搜索步骤, 使不同变体间 gadget 分布产生差异, 增加攻击者的搜索成本; 软件多样性可改变攻击面内可搜索到的 gadget 类别, 即攻击者潜在可获得的计算能力。

2) 构建 gadget 攻击链步骤: 不同 gadget 成

功构建可用攻击链的难度不同, 即各 gadget 执行需满足不同的特定条件, 软件多样性可通过改变 gadget 内部结构等方式使 gadget 构建攻击链的难度产生差异。

基于上述分析, 每个程序 P 对应的机器码 B 提取到的 gadget 特征空间可用三元组表示为 $\langle Q, C, D \rangle$, 其中 Q 为 B 中构建可用 gadget 攻击链的难度, C 为 B 中攻击者可获得的计算能力, D 为 B 中 gadget 在内存空间中的分布。

通过对比原始程序特征空间 $\langle Q, C, D \rangle$ 和变体程序特征空间 $\langle Q', C', D' \rangle$ 的差异, 实现软件多样性评估, 某一软件多样化技术有效的条件可被定义为

$$(Q' > Q) \vee (C' < C) \vee (|D' - D| > 0) \quad (1)$$

软件多样化技术带来的安全增益被定义为

$$\text{Gain}(\text{div}) = \lambda(Q' - Q) + \omega(C' - C) + \theta |D' - D| \quad (2)$$

其中, λ 、 ω 、 θ 为一组可调的权重值, 通过设置不同的大小, 从不同方面评估软件多样化技术。

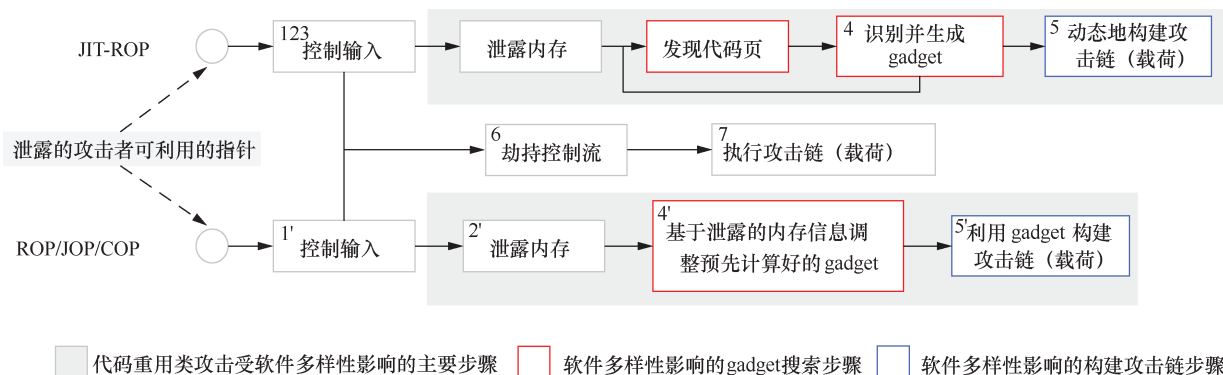


图 2 代码重用类攻击的流程
Figure 2 The workflow of code reuse attacks

2.2 基于 gadget 的度量指标

本节基于 gadget 特征提取进行的分析, 提出 gadget 质量、实用性和分布差异 3 个度量指标对应特征空间 $\langle Q, C, D \rangle$, 全面地量化软件多样性对代码重用攻击的影响。

2.2.1 gadget 质量

gadget 质量通过量化每个 gadget 构建攻击链的难易程度进行评估。

攻击者使用多个 gadget 构建攻击载荷时, 其主要目的是使用每个 gadget 中的特定功能指令, 然而, 大多数 gadget 除了包含攻击者所需的核心功能指令外, 还包含多条无用指令, 且其通常会

对整个基于 gadget 的代码重用攻击产生副作用, 即增加额外的限制条件。例如, 无用指令写入运行时才计算的内存地址, 在攻击过程中必须首先确保该内存地址可写, 才能保证攻击的实施; 无用指令修改核心功能指令中使用的寄存器 (“MOV EAX, EDX; ADD EAX, 0X10; RET;”) 等。

表 1 列出了根据 ROP、JOP 和 COP 攻击的差异, 以及 gadget 中无用指令的不同副作用所构建的评分标准, 该评分标准完善了文献[27]提出的针对 ROP 类 gadget 的质量分级规则, 并增加了条件分支等指令带来的副作用影响评分等。

评分标准的设置综合考虑 gadget 中的指令信

表1 包含无用指令的 gadget 指令评分标准
Table 1 The quality scoring standard for gadgets containing useless instructions

无用指令描述	评分
在寄存器上, 执行数据传送/算术/移位/循环移位操作	0.5
栈指针寄存器执行 pop 指令 (仅 ROP)	1
在 gadget 所操作的值目标寄存器上, 执行数据传送或算术操作	1
在 gadget 所操作的值目标寄存器上, 执行移位或循环移位操作	1.5
在栈指针寄存器上, 执行数据传送或算术操作 (仅 ROP)	2
在间接跳转目标寄存器/间接调用目标寄存器上, 执行数据传送/算术/交换操作 (仅 JOP/COP)	2
gadget 中包含 leave 指令 (仅 ROP) /条件分支指令/栈顶指针偏移量为负 (仅 ROP)	2
在栈指针寄存器上, 执行移位或循环移位操作 (仅 ROP)	3
在间接跳转目标寄存器/间接调用目标寄存器上, 执行移位或循环移位操作 (仅 JOP/COP)	3
栈指针寄存器是传送指令 (mov) 或交换指令 (xchg) 的目标寄存器 (仅 ROP)	4

息和 gadget 对栈指针寄存器偏移量的影响两个方面。在 gadget 中的指令信息方面, 主要依据数据传送类指令、算术类指令、移位和循环移位类指令对栈指针寄存器、gadget 操作的值目标寄存器以及其余各类寄存器由大到小的破坏性影响设置相应的质量分数, 对于 JOP 和 COP 类 gadget, 栈指针寄存器被替换为相应的间接跳转/调用目标寄存器, 此外, 条件分支类指令会增加各类 gadget 在使用上额外的限制条件, 所以对含有条件分支类指令的 gadget 也设置相应的质量分数。在 gadget 对栈指针寄存器偏移量的影响方面, 由于该偏移量主要影响 ROP 类 gadget 指向后续所需 gadget, 所以此方面仅涉及 ROP 类 gadget 的质量, 其中, “leave” 指令对栈指针寄存器偏移量的影响难以静态分析, 且若 gadget 中栈指针寄存器偏移量累计为负, 则栈指针难以被用于指向下一个 gadget, 所以分别设置对应上述两类情况的 gadget 相应的质量分数, 而 “mov” 指令和 “xchg” 指令对栈指针寄存器偏移量的影响相较于移位和循环移位类指令更加难以恢复, 所以设置包含该情况的 gadget 最大质量分数。综上, gadget 中无用指令的副作用越大, 攻击者使用其成功实施攻击的受限越多, 难度越高, 则设置其评分越高, gadget 中若包含多个上述各类情况, 则将相应的质量分数累加。

整个软件对应的 gadget 质量需首先计算软件中包含无用指令的可用 gadget 数量, 然后通过表 1

中的评分标准对相应 gadget 的质量进行分数累加计算, 从而量化攻击面构建攻击链的难度。

基于质量分数分别计算各多样化技术的平均质量变化率和平均绝对变化率。平均质量变化率需考虑各多样化技术产生的正向或负向质量影响, 平均绝对变化率仅考虑质量的绝对变化量, 不考虑正负影响, 从而分别从多样化技术对软件变体本身的安全质量影响和多样化技术引入各软件变体间攻击面质量的差异两个角度对软件多样性进行评估。

2.2.2 gadget 实用性

gadget 实用性通过量化二进制文件中存在的各个特定 microgadget (一种特定长度的 gadget 类型) 集合的计算能力进行评估。

评估软件多样性对构建图灵完备 gadget 集^[28] (包含图灵完备操作的 gadget 集合, 可执行任意计算行为) 的影响是一种普遍应用的方式, 但这会低估攻击者的能力, 即攻击者实施某些特定攻击并不需要完全得到图灵完备 gadget 集的计算能力, 仅需收集到所需的 gadget 就可成功实施攻击。

代码重用攻击主要用于一个完整漏洞攻击的初始阶段, 如通过实施攻击调用系统内核函数 (mmap 等) 分配具有可写可执行权限的内存空间, 并将攻击载荷副本复制至该内存空间, 通过间接分支指令将控制流重定向到该内存空间, 执行攻击载荷^[29]。对上述攻击场景而言, 图灵完备 gadget 集所提供的大量操作是多余的。

实际攻击中可通过 gadget 构建攻击,文献[30]中提出 2 byte 和 3 byte 长度的 microgadget,如“XCHG EAX、reg”、“XOR EAX、EBX”等,其数量大且具有图灵完备性,更加易于获得。因此,本文从实际攻击场景角度出发,构建 3 种具有不同计算能力的 microgadget 集合,其计算能力由低至高分别是 no-ASLR 集合、ASLR-proof 集合和图灵完备 microgadget 集,其中, no-ASLR 集合和 ASLR-proof 集合分配可执行内存空间,在该内存空间复制并执行攻击载荷,且 ASLR-proof 集合可在部署 ASLR 的系统中成功实施攻击。表 2 中列出了各集合的 microgadget 种类的数量信息。依据表 2 可定义每个程序 P 对应的机器码 B 中攻击者可获得的最大计算能力为

$$\begin{aligned} \text{Compute_power(B)} = \\ (\text{Count_noASLR(B)}, \\ \text{Count_ASLRproof(B)}, \text{Count_TC(B)}) \end{aligned} \quad (3)$$

其中,式(3)等号右侧的三元组为 3 种 microgadget 集合在 B 中对应的 microgadget 类别数量。

表 2 microgadget 集合信息
Table 2 Microgadget sets' information

microgadget 集合	Microgadget 类别数量
no-ASLR	11
ASLR-proof	35
图灵完备 microgadget 集	17

本文直接扫描文件中的 microgadget,并将其第一个指令作为判断其所属类别的标准。由于在 3 个集合中任何一类 microgadget 的缺失都会削弱整个集合的计算能力,所以本文通过计算部署多样化技术前后文件中 3 个集合的 microgadget 类别

数量的变化量,量化软件多样性对文件中存在的各个 microgadget 集合的计算能力的影响,以及软件变体间的差异。

2.2.3 gadget 分布差异

软件多样化技术作为一种提供概率保护的安全技术,其核心思想是使攻击难复现,攻击经验难复用,增加攻击成本。因此,本文基于核密度估计得到特定 gadget 集合在文件中空间分布的概率密度函数,比较多样化技术部署前后特定 gadget 集合在文件中分布的差异,评估软件多样性带来的安全增益,即空间分布差异越大,软件变体间同质性越低,攻击者对不同软件变体实施攻击的成本越大。

本文使用两种 gadget 集合进行分布差异的评估,分别是 MOV 图灵完备 gadget 集合和常用 gadget 集合。其中,前者是 MOV 指令构成的 gadget 集合,表 3 中列出了该集合的 gadget 类别信息,引入该集合是由于文献[31]仅通过 MOV 指令进行组合模拟实现图灵机,证明 MOV 指令具有图灵完备性,即可仅通过文件中相应 MOV 类 gadget 实现任意图灵操作;后者由统计 Metasploit 中部署在真实场景的 ROP 链中 10 个最常用的 gadget 类构成,表 4 中列出了该集合中各常用的 gadget 类别,除栈迁移(stack pivot)操作外,其余 gadget 仍属于图灵完备 gadget 集合。

上述两种集合在文件地址空间中的分布是通过核密度估计计算其概率密度函数实现的。核密度估计是一种估计随机变量概率密度函数的非参数估计方法,不需要任何关于分布函数的先验知识,完全由样本数据本身的分布决定,以拟合得到准确的结果。其主要思想如下:假设 $x_1, x_2, \dots, x_n$

表 3 MOV 图灵完备 gadget 集合
Table 3 MOV Turing complete gadget set

gadget 核心功能	gadget
寄存器赋值(由指定寄存器或立即数)	mov reg, reg/const
将某寄存器中的值存储至指定内存地址	mov [reg], reg
将某寄存器中的值或立即数存储至指定内存地址的 offset 偏移量处	mov [reg+offset], reg/const
将立即数存储至指定内存地址	mov [reg], const
将某内存地址中的值加载至指定寄存器	mov reg, [reg]
将某内存地址的 offset 偏移量中的值加载至指定寄存器	mov reg, [reg+offset]
调用系统内核函数	syscall

表 4 Metasploit 中常用的 gadget 类别
Table 4 Commonly used gadgets with gadget types in Metasploit

gadget 类别	功能	gadget	图灵完备 gadget 集
加载寄存器操作	将某数值加载至指定寄存器中	pop reg; ret pop reg; pop reg	是
算术操作	执行寄存器与某常数值加法运算	add reg, const	是
加载内存操作	将某内存地址中的值加载至指定寄存器	mov reg, [reg]; ret	是
无条件跳转操作	设置指令寄存器 EIP, 进行跳转	jmp reg	是
存储内存操作	将某寄存器中的值存储至指定内存地址	mov [reg], reg; ret	是
栈迁移操作	设置栈指针寄存器	xchg rsp, reg	否
逻辑操作	执行异或逻辑运算	xor reg, reg xor reg, const	是
寄存器赋值操作	通过其他寄存器/常数为指定寄存器赋值	mov reg, reg mov reg, const	是
函数调用操作	通过寄存器跳转至某函数	call reg mov reg, reg; call reg	是
系统调用操作	调用系统内核函数	syscall	是

是一维变量 X 中一组独立同分布的样本数据, 则 X 的未知概率密度函数 $f(x)$ 可被估计为

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x-x_i}{h}\right) \quad (4)$$

其中, $K(\cdot)$ 为核函数, h 为核函数的带宽, n 为样本数据的大小。本文将两个集合在文件中的地址信息作为样本数据, 使用高斯核函数进行核密度估计, 计算 gadget 分布的概率密度函数。

2.3 软件多样性评估算法

基于 gadget 度量指标的软件多样性评估算法如算法 1 所示, 其可量化不同多样化技术带来的安全增益。

算法 1 基于 gadget 度量指标的软件多样性评估算法

输入 原始程序 P , 需评估的多样化技术 Div_tech

输出 多样化技术 Div_tech 带来的安全增益 $Gain$

```

1) FUNCTION GET_FEATURE(B)
2)  $Q=0$ ;  $C=(0, 0, 0)$ ;  $D=(0, 0)$ ;
3)  $Loc\_mov, Loc\_common=[]$ ;  $j, k=0$ ;
4) FOR gadgeti IN search_gadget(B) THEN
5)    $Q=Q+Quality(gadget_i)$ ;
6)   IF gadgeti ∈ noASLR THEN
```

```
7)  $C[0] += 1$ ; //此前未搜到该类别
```

```
8)   ELSE IF gadgeti ∈ ASLRproof THEN
```

```
9)  $C[1] += 1$ ; //此前未搜到该类别
```

```
10)  ELSE IF gadgeti ∈ TC THEN
```

```
11)  $C[2] += 1$ ; //此前未搜到该类别
```

```
12)  END IF
```

```
13)  IF gadgeti ∈ mov_set THEN
```

```
14)  $Loc\_mov[j++] = gadget_i$  的地址;
```

```
15)  ELSE IF gadgeti ∈ common_set
```

```
THEN
```

```
16)  $Loc\_common[k++] = gadget_i$  的地址;
```

```
17)  END IF
```

```
18) END FOR
```

```
19)  $D=(KDE(Loc\_mov), KDE(Loc\_common))$ ;
```

```
20) RETURN  $\langle Q, C, D \rangle$ 
```

```
21) END FUNCTION
```

```
22)  $B=Compile(P)$ ,  $B'=Div\_tech(P)$ ;
```

```
23)  $\langle Q, C, D \rangle = GET\_FEATURE(B)$ ;  $\langle Q', C', D' \rangle = GET\_FEATURE(B')$ ;
```

```
24)  $Gain(Div\_tech) = \lambda(Q' - Q) + \omega(C' - C) + \theta(D' - D)$ ;
```

第 1) 行~第 21) 行是提取机器码文件 B 中 gadget 特征, 实现各 gadget 度量指标计算的函数, 其中, 第 2)、3) 行初始化各度量指标值, 第 5) 行累加各

gadget 质量, 第 6)行~第 12)行搜集属于 3 个 microgadget 集合的 microgadget, 并记录各集合中 microgadget 类别数量, 根据式(3)为特征空间中攻击者可获得的最大计算能力 C 赋值, 第 13)行~第 17)行记录属于特定集合的 gadget 在文件中的地址, 第 19)行通过核密度估计计算特定集合 gadget 分布的概率密度函数, 第 20)行函数返回机器码文件 B 对应的各 gadget 指标值; 第 22)行得到特定多样化技术部署前后的机器码; 第 23)行调用第 1)行~第 21)行所实现的函数 GET_FEATURE, 分别得到多样化前后机器码文件 B、B' 的 gadget 特征空间对应的 gadget 指标值; 第 24)行依据得到的 gadget 指标值, 按式(2)计算多样化技术 Div_tech 带来的安全增益, 其中 λ 、 ω 、 θ 值可调。

3 实验和分析

本节对各粒度的软件多样化技术进行实验评估, 即采用多样化技术所包含的具体方法对源码文件进行处理, 然后按 2.3 节中的软件多样性评估算法对得到的机器码进行 gadget 特征提取, 计算各粒度软件多样化技术带来的安全增益。

3.1 实验配置

软件多样性评估的实验环境为 Ubuntu 18.04 LTS 操作系统。

(1) 数据集

本文使用 GNU 核心工具组 (coreutils 8.2) 作为多样化技术处理的应用程序集, 并将常被用作攻击演示的应用程序作为标准, 使用开源的 bzip2、nginx、hiawatha、mysql 和 mupdf 进行实验和评估。表 5 中列出了各应用程序 (集) 信息。

表 5 应用程序 (集) 信息
Table 5 Application (set) information

应用程序 (集)	应用描述
GNU coreutils 8.2	GNU 操作系统的核心实用程序 (101 个应用程序)
bzip2 v1.0.8	压缩软件
nginx v1.8.1	Web 服务器
hiawatha v9.0	Web 服务器
mysql v5.5	数据库
mupdf v0.9	PDF 阅读器

(2) 多样化工具

根据多样化对象的粒度差异^[6], 可将现有软件多样化技术细化为指令级^[32]、基本块级^[33]、函数级^[34]和程序级等。本文使用 MCR^[9,15]和 OLLVM^[35]工具提供不同粒度的软件多样化技术, 其中各粒度的多样化技术包含以下功能。

1) 指令级: 通过指令替换或指令重排序实现, 包含寄存器随机化、NOP 插入和等效指令替换功能。

2) 基本块级: 通过改变函数内基本块的数量和排序实现, 包含基本块分割和伪控制流功能。伪控制流技术主要由基本块插入和不透明谓词等技术实现。

3) 函数级: 包含栈布局随机化和控制流扁平功能。栈布局随机化由栈帧填充和栈帧布局洗牌等实现。控制流扁平通过设置特定的跳转表, 将基本块间的直接跳转替换为间接跳转, 使控制流程图更为复杂。

4) 程序级: 包含函数随机化功能, 可分别在编译和链接时进行函数重排序。

(3) 评估方法实现

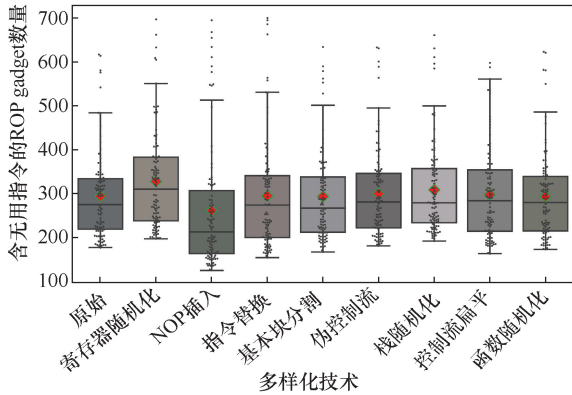
本文利用开源 gadget 查找工具 ROPgadget, 对其进行修改, 通过正则表达式识别等方式实现 gadget 质量的计算, 以及各类 gadget 集合的信息搜集。gadget 度量指标的权重值(λ , ω , θ)设置为(1, 0, 0), (0, 1, 0)和(0, 0, 1), 分别从 gadget 质量、实用性和分布差异 3 个角度进行软件多样性评估。

3.2 gadget 质量

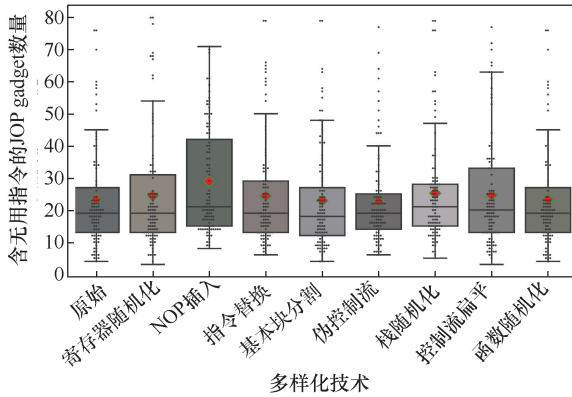
图 3(a)~图 3(c)分别对应多样化技术处理后, 文件中包含无用指令的 ROP gadget、JOP gadget 和 COP gadget 数量, 每个数据点代表对一个文件扫描所得的数量, 红色表示 106 个文件的均值。

由图 3(a)可知, 指令级多样化技术会使 ROP gadget 数量有较大的变化, 即增加较大的不可预测性, 验证了多样化技术对攻击成本增益的有效性。其中, 寄存器随机化使含无用指令的 ROP gadget 数量大幅增加, 主要是其对指令中的寄存器进行随机分配, 产生额外的指令, 使 gadget 利用复杂。此外, 基本块分割和控制流扁平化也使含无用指令的 ROP gadget 数量有较大变化, 但其

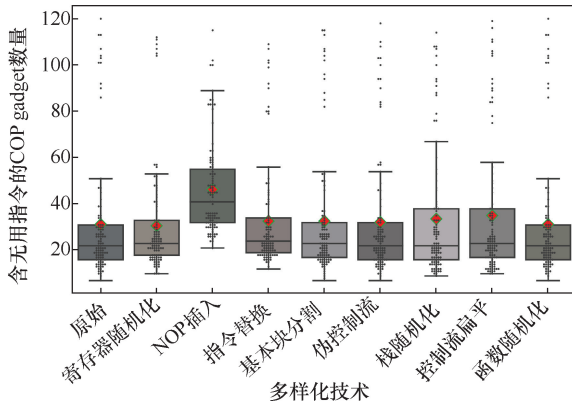
均值与部署前差异较小, 主要是上述两种多样化技术使控制流图发生大幅度改变, 提高了同一软件间差异的不可预测性。



(a) 包含无用指令的ROP gadget数量



(b) 包含无用指令的JOP gadget数量



(c) 包含无用指令的COP gadget数量

图3 含无用指令的可用 gadget 数量
Figure 3 The number of gadget containing useless instructions

图 3(b)、图 3(c)中含无用指令的 JOP gadget 和 COP gadget 数量相较于 ROP gadget 有大幅下降, 主要是 JOP 和 COP gadget 本身在软件中数量较少。指令级多样化技术对两者影响较大, 但其中的等效指令替换对 JOP gadget 和 COP gadget

影响不大, 引起的差异较小。由于控制流扁平化会大量增加“JMP”等跳转指令, 所以会使可用 JOP gadget 数量增加; 由于栈布局随机化涉及堆栈转换, 需要调用 malloc(·) 为变量分配内存, 使可用 COP gadget 数量增加。

表 6 中列出了各类 gadget 的质量分数及变化率。其中, NOP 插入对 gadget 质量有较大的影响, 对 COP gadget 造成了 33.44% 的质量下降, 使同一软件的变体间攻击面质量产生较大的差异, 这主要是 MCR 工具的 NOP 插入实现中不是仅插入“NOP”指令, 而是随机插入“MOV ESP, ESP”或“LEA ESI [ESI]”等 5 种与 NOP 等效的指令^[15], 以增加不可预测性, 对质量评分产生较大影响, 且会额外引入非预期 gadget^[2], 即对原始指令边界以外的任意偏移进行解码, 从而得到潜在的 gadget 指令序列。

通过表 6 的平均绝对变化率可知, 软件多样化技术能够使攻击面质量产生较大的变化, 从而增加攻击者实施大规模攻击的成本, 其中指令级多样化技术通常能导致更大的变化, NOP 插入最为显著, 由于控制流扁平化等技术的部署会产生大量跳转指令, 对 JOP gadget 有较大的影响, 函数随机化对质量的影响最小, 因为其仅以函数为单位对程序进行重排列。但综合分析表 6 的平均质量变化率可知, 部分软件多样化技术在降低软件同质性的同时, 会导致软件攻击面质量下降, 其中, 指令级多样化技术造成的质量下降较大, 从而降低攻击者成功构建可用的代码重用攻击链的难度。

3.3 gadget 实用性

图 4 为各粒度的软件多样化技术对 3 个 microgadget 集合中各类 microgadget 的影响。其中灰色虚线为原始文件中 3 个集合的 microgadget 类别数量, 并以此为基准比较多样化技术部署前后的差异, 每个数据点为数据集中 106 个文件在相应集合下 microgadget 类别数量的平均值。

由图 4 可知, 存在部分应用程序的 3 个 microgadget 集合均未找到全部类别的 microgadget。指令级多样化技术显著扰乱内存布局, 能够对各软件变体产生较大的影响, 使软件变体中各集合的 microgadget 类别数量与原始文件产生较大差异。其中, NOP 插入能够显著减少

表6 不同多样化技术处理下 ROP/JOP/COP gadget 质量分数及变化率
Table 6 Quality score and change rate of ROP/JOP/COP gadget under different diversity technology

攻击类型	ROP	JOP	COP	平均质量变化率	平均绝对变化率
原始	1.5705	1.3994	1.6971		
寄存器随机化	1.4571 (-7.22%)	1.3005(-7.07%)	1.5501 (-8.66%)	-7.65%	7.65%
Nop 插入	1.4164 (-9.81%)	1.6119(15.19%)	1.1296(-33.44%)	-9.35%	19.48%
指令替换	1.5124 (-3.7%)	1.2492(-10.73%)	1.6228(-4.38%)	-6.27%	6.27%
基本块分割	1.5751 (0.29%)	1.3562(-3.09%)	1.6837(0.79%)	-1.20%	1.39%
伪控制流	1.5785 (0.51%)	1.3305(-4.92%)	1.6654(-1.87%)	-2.09%	2.43%
栈布局随机化	1.5846 (0.90%)	1.2546(-10.35%)	1.6767(-1.20%)	-3.55%	4.15%
控制流扁平	1.6902 (7.62%)	1.2603(-9.94%)	1.742(2.65%)	0.11%	6.74%
函数随机化	1.5796 (0.58%)	1.3861(-0.95%)	1.7035(0.38%)	0.00%	0.64%

microgadget 类别数量,有效降低软件本身可用 microgadget 的计算能力,主要是 NOP 插入 1 byte 或 2 byte 长度的指令破坏了 microgadget 结构或使单个 microgadget 复杂,而 microgadget 具体功能的识别是通过第一个操作码进行的,减少了可识别到的 microgadget 类别,从而增加了攻击者找到所需 microgadget 的难度。

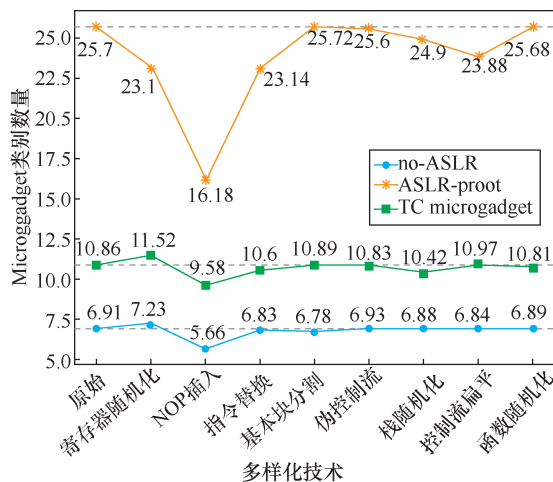


图4 多样化技术处理下3个 microgadget 集合中 microgadget 类别数量
Figure 4 The number of microgadget types in three microgadget sets under diversity technology

基本块级和函数级的多样化技术对内存的扰动,以及 microgadget 的影响相对于指令级多样化技术较小;而程序级的函数随机化仅会使 microgadget 位置发生移动,不会额外产生和破坏 microgadget,所以部署函数随机化后 3 个不同计算能力的 microgadget 集合中 microgadget 类别数量近似于原始文件。

综上,软件多样化技术可干扰内存布局、破坏和改变 microgadget 结构,也可使同一应用程序的不同变体间可利用的 microgadget 信息存在一定的差异,从而达到攻击难复现,攻击经验难复用的效果,有效增加攻击成本。部分细粒度的多样化技术可使应用程序中可识别到的 microgadget 类别数量减少。

3.4 gadget 分布差异

本节对 mysql v5.5 部署多样化技术,并依据特定 gadget 集合的 gadget 类别信息,通过正则表达式扫描相应的 gadget,基于核密度估算 gadget 分布,评估不同的多样化技术对两种特定 gadget 集合分布的影响,并对 gadget 在文件中各段的位置进行分析。

图 5(a)、图 5(b)分别为.text 段中 Metasploit、MOV 图灵完备 gadget 集合的核密度估计分布,横轴的轴须图为各个 gadget 的位置,横坐标标将.text 段的长度归一化至[0, 1]。由图 5(a)、图 5(b)可知,多样化技术对两个集合分布的影响基本一致,但其分布存在一定的差异,且 MOV 图灵完备 gadget 在.text 段中较少。NOP 插入、伪控制流和基本块分割能显著地改变.text 段中 gadget 分布,其中 NOP 插入以一定的概率在每个指令后插入固定数量的多种 NOP 等效指令,使多样化技术部署后每条指令相较于原始文件有较大的偏移,从而改变特定 gadget 集合的分布,增加攻击者重复分析的成本;伪控制流插入基本块,并使用不透明谓词将控制流重定向至原始基本块,会导致

原本的 gadget 产生较大的偏移。但上述 3 种多样化技术都会使文件大小有一定的增长。

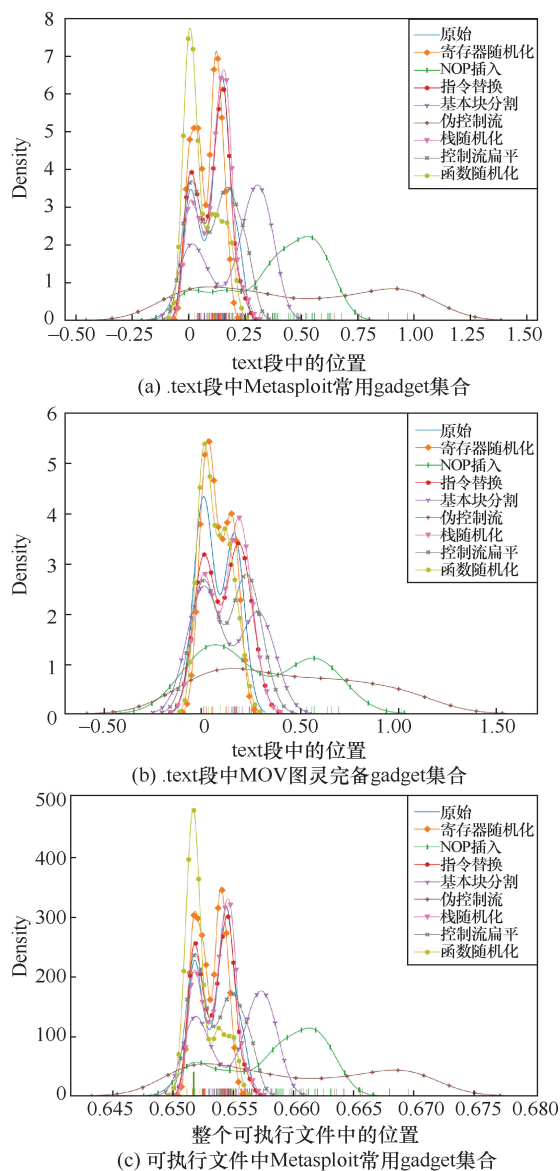


图5 多样化技术部署前后特定集合的 gadget 分布
Figure 5 Gadget distribution for specific sets before and after diversity technology deployment

寄存器随机化和函数随机化可对 gadget 分布产生一定程度的影响。函数随机化以函数为单位改变 gadget 分布, 使 gadget 位置产生偏移。其余多样化技术上对 gadget 位置改变的影响较小, 不能使 gadget 集合的分布产生较大差异。

图5为Metasploit常用gadget集合在.text段和整个可执行文件中的分布, 图5(c)的轴须图中较长部分为可执行文件中.text段的起始地址, 除

函数随机化外, 其余多样化技术.text段的起始地址相同。绝大多数 gadget 位于.text段中, 但仍有极少量 gadget 位于其他段中, 如.init段和.fini段等。由于函数随机化可在编译或链接时进行, 会对.init段前的.gnu.hash段和.dynsym段产生改变, 从而使.init段中的 gadget 产生偏移。本文使用的其余多样化技术不会对上述.init段和.fini段中的crt.o等文件进行处理, 所以其中的 gadget 不受多样化技术部署的影响(由于.text段长度变化, 位于其后的.fini段中的 gadget 会产生一定偏移), 则.init段中 gadget 信息可被攻击者重用, 降低攻击成本, 产生一定的负向影响。

综上, 通过评估多种不同粒度的软件多样化技术对特定 gadget 集合分布的影响可知, 对内存布局或源码结构有较大影响的多样化技术能够使 gadget 分布有较大的差异, 从而增加攻击不同软件变体的分析成本等, 但满足这种特性的方法通常会导致软件膨胀, 造成额外的存储开销。本文实验评估中使用的多样化技术仅对.text段进行处理, 而实际上 gadget 可分布在.init段等不被多样化技术处理的段中, 可导致对软件变体的攻击成本下降。通过对MOV图灵完备 gadget 集合分布进行分析可知, 该集合中各类 gadget 广泛存在于软件中, 可被攻击者利用实施恶意行为。

3.5 评估总结

细粒度的多样化技术能使软件中大量的 gadget 重定位/修改/移除, 降低软件同质性, 从而攻击者需要重新扫描每个变体的 gadget, 定制恶意载荷实施攻击, 增加了攻击成本。

由 gadget 质量的实验评估结果可知, 细粒度的多样化技术可有效地使变体间存在较大的差异性, 但其中存在的部分多样化技术会导致生成的软件变体质量评分下降, 即变体相较于原始程序更容易被攻击者利用以构建所需的攻击链。未来可依据本文提出的 gadget 质量评分标准中使构建攻击链难度提升的相关指令设计软件多样化技术, 以增加攻击者成功构建攻击链的难度。

由 gadget 实用性的实验评估结果可知, 部分细粒度的多样化技术可通过破坏 microgadget 结构使软件中可识别的 microgadget 类别数量减少。未来可通过将软件多样化技术和去膨胀技术^[36]

结合, 删除不影响程序运行的代码, 减少应用程序中的 **microgadget** 类别数量, 从而减小攻击者可获得的最大计算能力。

由 **gadget** 分布差异的实验评估结果可知, 引起较大分布差异的多样化技术往往会导致额外的存储开销; 除函数随机化外, 其他多样化技术无法对存在可用 **gadget** 的动态库等进行处理。未来可通过将多样化技术与去膨胀技术^[36]结合以缓解额外的存储开销, 为了加强对动态库中函数调用的保护, 可将函数定制^[37]、影子栈等技术与软件多样化技术结合以实现更高安全性的防御机制。

4 结束语

本文提出了一种基于 **gadget** 特征分析的软件多样性评估方法, 包括 **gadget** 质量、实用性和分布 3 个指标, 通过计算相应的 **gadget** 质量评分、**microgadget** 集合计算能力和特定 **gadget** 集合分布差异全面地度量了软件多样性对代码重用类攻击各攻击步骤的影响程度, 从而有效评估了软件多样性所带来的安全增益。通过对多种软件多样化技术进行评估实验, 验证了评估方法的有效性, 依据得到的评估结果, 分析和讨论了现有多样化技术的有效性和不足, 为提出更具安全性的软件多样化技术提供量化评估决策支撑。

参考文献:

- [1] BIRMAN K P, SCHNEIDER F B. The monoculture risk put into context[J]. *IEEE Security & Privacy*, 2009, 7(1): 14-17.
- [2] SHACHAM H. The geometry of innocent flesh on the bone: return-into-libc without function calls (on the x86)[C]//*Proceedings of the 14th ACM Conference on Computer and Communications Security*. 2007: 552-561.
- [3] CARLINI N, WAGNER D. ROP is still dangerous: breaking modern defenses[C]//*Proceedings of 23rd USENIX Security Symposium (USENIX Security 14)*. 2014: 385-399.
- [4] CHECKOWAY S, DAVI L, DMITRIENKO A, et al. Return-oriented programming without returns[C]//*Proceedings of the 17th ACM Conference on Computer and Communications Security - CCS '10*. 2010: 559-572.
- [5] SNOW K Z, MONROSE F, DAVI L, et al. Just-in-time code reuse: on the effectiveness of fine-grained address space layout randomization[C]//*Proceedings of 2013 IEEE Symposium on Security and Privacy*. 2013: 574-588.
- [6] LARSEN P, HOMESCU A, BRUNTHALER S, et al. SoK: automated software diversity[C]//*Proceedings of 2014 IEEE Symposium on Security and Privacy*. 2014: 276-291.
- [7] BAUDRY B, MONPERRUS M. The multiple facets of software diversity[J]. *ACM Computing Surveys*, 2015, 48(1): 1-26.
- [8] PAX T. PaX address space layout randomization (ASLR)[R]. 2003.
- [9] CRANE S, LIEBCHEN C, HOMESCU A, et al. Readactor: practical code randomization resilient to memory disclosure[C]//*Proceedings of 2015 IEEE Symposium on Security and Privacy*. 2015: 763-780.
- [10] YAO D D, SHU X K, CHENG L, et al. Anomaly detection as a service: challenges, advances, and opportunities[J]. *Synthesis Lectures on Information Security, Privacy, and Trust*, 2017, 9(3): 1-173.
- [11] GRIER C, BALLARD L, CABALLERO J, et al. Manufacturing compromise: the emergence of exploit-as-a-service[C]//*Proceedings of the 2012 ACM Conference on Computer and Communications Security*. 2012: 821-832.
- [12] GEARHART A S, HAMILTON P A, COFFMAN J. An analysis of automated software diversity using unstructured text analytics[C]//*Proceedings of 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. 2018: 79-80.
- [13] LE Q, MIKOLOV T. Distributed representations of sentences and documents[C]//*Proceedings of International Conference on Machine Learning*. 2014: 1188-1196.
- [14] COFFMAN J, KELLY D M, WELLONS C C, et al. ROP gadget prevalence and survival under compiler-based binary diversification schemes[C]//*Proceedings of the 2016 ACM Workshop on Software PROtection*. 2016: 15-26.
- [15] HOMESCU A, NEISIUS S, LARSEN P, et al. Profile-guided automated software diversity[C]//*Proceedings of the 2013 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. 2013: 1-11.
- [16] 熊浩, 晏海华, 郭涛, 等. 代码相似性检测技术: 研究综述[J]. *计算机科学*, 2010, 37(8): 9-14, 76.
- [17] XIONG H, YAN H H, GUO T, et al. Code similarity detection: a survey[J]. *Computer Science*, 2010, 37(8): 9-14, 76.
- [18] KRINKE J. Identifying similar code with program dependence graphs[C]//*Proceedings of Eighth Working Conference on Reverse Engineering*. 2001: 301-309.
- [19] BAKER B S. A program for identifying duplicated code[J]. *Computing Science and Statistics*, 1993: 49-49.
- [20] ENGELS S, LAKSHMANAN V, CRAIG M. Plagiarism detection using feature-based neural networks[C]//*Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*. 2007: 34-38.
- [21] ZHANG L, ZHUANG Y T, YUAN Z M. A program plagiarism detection model based on information distance and clustering[C]//*Proceedings of 2007 International Conference on Intelligent Pervasive Computing (IPC 2007)*. 2007: 431-436.
- [22] PRIYADARSHAN S, NGUYEN H, SEKAR R. Practical fine-grained binary code randomization[C]//*Proceedings of Annual Computer Security Applications Conference*, 2020: 401-414.
- [23] 刘镇武, 隋然, 张铮, 等. 基于信息熵与软件复杂度的软件多样性评估方法[J]. *信息工程大学学报*, 2020, 21(2): 207-213.
- [24] LIU Z W, SUI R, ZHANG Z, et al. Software diversity evaluation

- method based on information entropy and software complexity[J]. Journal of Information Engineering University, 2020, 21(2): 207-213.
- [23] BALACHANDRAN V, EMMANUEL S. Software protection with obfuscation and encryption[C]//Proceedings of Information Security Practice and Experience. 2013: 309-320.
- [24] BANESCU S, COLLBERG C, GANESH V, et al. Code obfuscation against symbolic execution attacks[C]//Proceedings of the 32nd Annual Conference on Computer Security Applications. 2016: 189-200.
- [25] KELLY D M, WELLONS C C, COFFMAN J, et al. Automatically validating the effectiveness of software diversity schemes[C]//Proceedings of 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S). 2019: 1-2.
- [26] AHMED S, XIAO Y, SNOW K Z, et al. Methodologies for quantifying (Re-) randomization security and timing under JIT-ROP[C]//Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. 2020: 1803-1820.
- [27] FOLLNER A, BARTEL A, BODDEN E. Analyzing the gadgets[C]//Proceedings of International Symposium on Engineering Secure Software and Systems. 2016: 155-172.
- [28] ROEMER R, BUCHANAN E, SHACHAM H, et al. Return-oriented programming: systems, languages, and applications[J]. ACM Transactions on Information and System Security, 2012, 15(1): 1-34.
- [29] DAI Z D. Practical return-oriented programming[J]. Source Boston, 2010.
- [30] HOMESCU A, STEWART M, LARSEN P, et al. Microgadgets: size does matter in turing-complete return-oriented programming[J]. WOOT, 2012, 12: 64-76.
- [31] DOLAN S. MOV is turing-complete (2013)[EB].
- [32] HAWKINS W H, HISER J D, CO M, et al. Zipr: efficient static binary rewriting for security[C]//Proceedings of 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). 2017: 559-566.
- [33] KOO H, CHEN Y H, LU L, et al. Compiler-assisted code randomization[C]//Proceedings of 2018 IEEE Symposium on Security and Privacy. 2018: 461-477.
- [34] CONTI M, CRANE S, FRASSETTO T, et al. Selfrando: securing the tor browser against de-anonymization exploits[J]. Proc Priv Enhancing Technol, 2016(4): 454-469.
- [35] JUNOD P, RINALDINI J, WEHRLI J, et al. Obfuscator-LLVM: software protection for the masses[C]//Proceedings of 2015 IEEE/ACM 1st International Workshop on Software Protection. 2015: 3-9.
- [36] PORTER C, MURURU G, BARUA P, et al. BlankIt library debloating: getting what You want instead of cutting what you don't[C]//Proceedings of PLDI 2020: Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation. 2020: 164-180.
- [37] MISHRA S, POLYCHRONAKIS M. Saffire: context-sensitive function specialization against code reuse attacks[C]//Proceedings of 2020 IEEE European Symposium on Security and Privacy (EuroS&P). 2020: 17-33.

[作者简介]



谢根琳 (1999-), 男, 河北辛集人, 信息工程大学硕士生, 主要研究方向为网络空间安全和软件多样化。



程国振 (1986-), 男, 山东菏泽人, 信息工程大学副教授, 主要研究方向为网络空间安全、云安全。



王亚文 (1990-), 男, 河南郑州人, 信息工程大学助理研究员, 主要研究方向为拟态防御和云计算。



王庆丰 (1994-), 男, 河南南阳人, 信息工程大学助理研究员, 主要研究方向为拟态防御和软件多样化。