

恶意代码动态分析对抗与反对抗技术综述

王晨阳^{1,2}, 彭国军^{1,2}, 杨秀璋³, 周逸林^{1,2}

(1. 空天信息安全与可信计算教育部重点实验室, 湖北 武汉 430072;

2. 武汉大学国家网络安全学院, 湖北 武汉 430072;

3. 贵州大学公共大数据国家重点实验室, 贵州 贵阳 550025)

摘要: 动态分析是恶意代码分析与检测的关键技术, 能够刻画样本在运行时的真实行为特征。在攻防博弈的过程中, 恶意代码采用动态分析对抗技术逃避分析, 安全研究人员则通过反对抗技术提升分析系统应对规避型恶意软件的能力。已有综述大多聚焦于攻防技术的设计与实现, 缺乏对其与恶意代码动态分析各核心环节之间关联的系统梳理。为此, 该研究以恶意代码动态分析的工作流程为主线, 从攻防对抗的视角出发, 对相关技术进行了系统综述。首先, 介绍了动态分析的典型工作流程; 其次, 总结了对抗技术的设计与实现方式, 包括恶意载荷隐藏、分析环境感知、行为捕获逃避及分析模型欺骗技术, 剖析其规避机制; 再次, 整理与归纳了相应的反对抗技术, 包括恶意载荷捕获、环境感知规避、行为触发与监控增强及分析模型增强技术, 揭示动态分析系统演变背后的攻防对抗逻辑; 最后, 探讨了恶意代码动态分析领域未来的发展趋势与重点研究方向, 以期为恶意代码分析与检测研究提供参考。

关键词: 恶意代码; 动态分析; 调试器; 沙箱; 对抗与反对抗

中图分类号: TP309.5

文献标志码: A

DOI:10.11959/j.issn.2096-109x.2025063

Survey on malicious code dynamic analysis evasion and anti-evasion techniques

WANG Chenyang^{1,2}, PENG Guojun^{1,2}, YANG Xiuzhang³, ZHOU Yilin^{1,2}

1. Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, Wuhan 430072, China

2. School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China

3. State Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China

Abstract: Dynamic analysis serves as a fundamental technique in malicious code analysis and detection, enabling the observation of real-time behaviors exhibited by malicious samples. In the ongoing adversarial arms race, malicious code employs dynamic analysis evasion techniques to evade detection, while defenders develop countermeasures to enhance the capability of dynamic analysis systems to cope with evasive malware. Existing surveys predominantly focus on the design and implementation of individual offensive and defensive techniques, lacking a comprehensive examination of their integration with the core stages of dynamic malicious code analysis. To address this limitation, a systematic survey of evasion techniques and anti-evasion techniques was conducted from an adversarial perspective, structured around the dynamic analysis workflow. First, the typical workflow of malicious code dynamic analysis was introduced. Next, the design and implementation methods of evasion techniques were

收稿日期: 2025-04-30; 修回日期: 2025-05-24

基金项目: 国家自然科学基金(62172308, 61972297, 62172144, 62562012); 贵州省基础研究计划(MS[2025]686)

Foundation Items: The National Natural Science Foundation of China (62172308, 61972297, 62172144, 62562012), Guizhou Provincial Basic Research Program (MS[2025]686)

summarized, including malware payload hiding, environment detection, behavior capture evasion, and analysis model deception, and their evasion strategies were analyzed. Then, the corresponding countermeasures were reviewed, including malicious payload extraction, evasion-resistant environment construction, behavior triggering and monitoring enhancements, and model hardening, and the adversarial interplay driving the evolution of dynamic analysis systems was revealed. Finally, the future development trends and key research directions in the field of dynamic analysis of malicious code were discussed, with the aim of providing references for the research on malicious code analysis and detection.

Keywords: malicious code, dynamic analysis, debugger, sandbox, evasion and anti-evasion

0 引言

随着计算机技术的不断发展, 恶意代码持续演化迭代, 恶意软件攻击的态势日趋严峻。当前, 恶意代码的种类与数量在快速增长, 木马、勒索软件、挖矿病毒、僵尸网络等传统恶意程序依然活跃, 新型恶意代码家族层出不穷^[1-3]。与此同时, 具有极强隐蔽性、对抗性和破坏性的 APT 攻击, 也常以恶意代码为载体执行信息搜集、数据窃取、持久化控制等攻击任务^[4]。在此背景下, 构建高效的恶意代码分析系统, 精准捕获恶意行为特征并实现软件恶意性检测, 已成为网络空间安全治理领域的重要研究课题。

程序动态分析技术通过捕获程序运行过程中的动态信息来解析其行为与特性, 目前已被安全研究领域广泛应用于恶意代码分析。其典型工作流程主要包含预处理、分析环境准备、样本执行与行为捕获、行为数据分析 4 个核心环节。在攻防对抗的持续博弈中, 恶意代码逐渐发展出动态分析对抗技术以规避恶意行为的捕获, 这类恶意软件被称为规避型恶意软件。攻击者通过无文件攻击等手段隐藏恶意载荷, 依托调试器特征与沙箱指纹识别分析环境, 借助时延、条件触发等机制逃避行为捕获, 并在恶意行为中注入对抗序列以欺骗分析模型, 最终形成覆盖动态分析全流程的对抗体系。

为增强动态分析系统对规避型恶意软件的检测能力, 研究人员开发了一系列反对抗技术。其一, 通过构建溯源图对系统行为进行综合分析, 以检测无文件攻击并捕获恶意载荷; 其二, 升级运行环境, 采用更高特权等级的监控方式构建透明分析环境, 避免被恶意代码感知到分析环境的存在; 其三, 利用时延跳过与路径探索技术触发隐藏的恶意功能, 同时增强行为监控的抗干扰能力; 其四, 提升分析模型的鲁棒性, 动态优化检

测规则, 以应对对抗样本带来的威胁。

目前, 动态分析对抗与反对抗技术已在恶意代码分析领域的部分综述中被提及^[5-6], 也有专门以此为主题的综述研究^[7-8]。但现有综述多局限于对攻防技术的孤立讨论, 研究视角相对单一, 尚未系统建立这些技术与动态分析各核心环节的关联。为此, 本文聚焦 Windows 平台的恶意代码分析, 以恶意代码动态分析工作流程为主线, 归纳相关攻防对抗技术, 揭示动态分析系统迭代升级的内在逻辑, 并展望未来的研究方向。

1 恶意代码动态分析概述

恶意代码动态分析的典型工作流程如图 1 所示, 具体可划分为以下 4 个核心阶段。

1) 预处理阶段: 对软件样本开展初步静态解析, 提取文件类型、API 导入表等基础特征, 为后续动态分析提供可执行代码; 同时生成系统环境要求、API Hook 监控点位等关键上下文信息。这一阶段的攻防博弈核心在于恶意载荷的隐藏与捕获。

2) 分析环境准备阶段: 根据软件样本的类型, 利用模拟器、虚拟机等工具构建隔离的运行环境; 结合分析需求选取调试器或沙箱等分析方式, 为样本运行搭建安全、可靠的分析场景。本阶段的攻防对抗焦点是分析环境的感知与反感知。攻击者通过环境感知技术和工具判断是否处于分析场景, 防护方则通过优化环境配置构建高透明度环境以规避感知。

3) 样本执行与行为捕获阶段: 通过运行软件样本记录其动态行为, 重点捕获样本运行过程中的指令序列、API 调用、网络通信、文件操作等行为数据, 为后续恶意性检测等分析任务提供核心数据支撑。此阶段的攻防博弈围绕行为隐藏与行为捕获展开。攻击者通过隐藏恶意行为、干扰

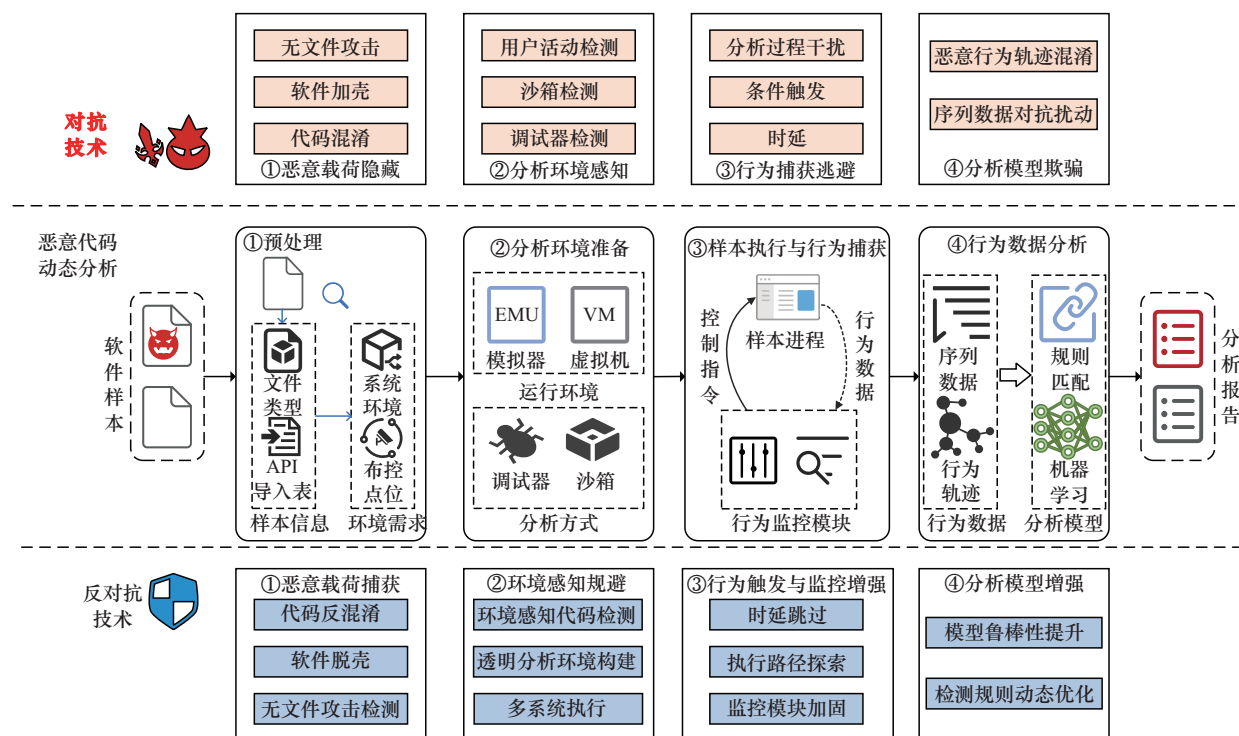


图1 恶意代码动态分析的典型的工作流程

分析来逃避行为捕获，防护方则通过触发隐藏行为、增强行为监控进行反制。

4) 行为数据分析阶段：对收集到的样本行为数据进行深度分析，采用规则匹配、机器学习、深度学习等方法评估样本的恶意性，并生成分析报告。该阶段的攻防对抗聚焦于模型欺骗与模型增强。攻击者通过构造对抗样本、伪造正常行为等方式欺骗分析模型，防护方则通过优化检测规则、提升模型鲁棒性来增强分析能力。

2 恶意代码动态分析对抗技术

本节围绕恶意代码动态分析的典型工作流程，系统介绍针对各分析环节的对抗技术。

2.1 恶意载荷隐藏

恶意载荷隐藏技术通过隐藏攻击载荷和可执行文件等方式对抗动态分析，其常用的手段主要有代码混淆、软件加壳及无文件攻击技术。

2.1.1 代码混淆

代码混淆是一种软件保护技术，它能够在保持程序功能不变的条件下，通过语义等价变换来隐藏执行逻辑和关键数据，从而有效阻止逆向工程，抵御动/静态分析^[9-10]。恶意软件常利用代码混淆技术隐藏恶意载荷，防止其被分析系统提取。

代码混淆可将程序源代码、中间表示或二进制文件作为输入，通过控制流平坦化等方式改写原始程序的控制流结构，通过不透明谓词注入、路径分支混淆^[11]等方式保护程序的跳转信息，并利用加密、变量重命名等手段保护程序数据与布局信息，最终生成具备抗分析能力的可执行实体。为应对符号执行等新型自动化分析技术的发展，代码混淆技术已从传统的单一语法层变换，逐步演进为线程间执行路径混淆^[12]、多项反混淆技术高效融合^[13]、代码虚拟化与闪存控制器软硬件结合^[14]等多维度协同混淆。同时，利用大语言模型生成混淆代码的尝试也已出现^[15]，进一步提升了混淆的隐蔽性。

恶意代码常利用代码混淆技术隐藏其执行逻辑和数据，这会对动态分析的多个环节造成干扰。在预处理阶段，控制流混淆技术会破坏反汇编和反编译过程（例如，难以依据混淆样本的控制流图还原原始执行逻辑），字符串加密操作则会隐藏样本程序的关键API调用信息，影响分析环境的准备；在样本执行与行为捕获阶段，混淆样本的分支路径过多可能导致“路径爆炸”，大幅降低分析效率。

2.1.2 软件加壳

软件加壳是恶意代码隐藏恶意载荷的另一核心技术。软件加壳是一种软件保护技术,被恶意代码广泛用于隐藏恶意载荷。加壳工具通过压缩、加密等方式隐藏可执行二进制文件的原始内容,并附加脱壳例程。加壳后的程序在运行时会先执行脱壳例程,待原始代码在内存中解密还原后再执行原始代码。为增强对抗分析的能力,软件加壳技术已从最初的简单壳(仅对原始代码进行压缩,执行时通过一次脱壳例程还原原始代码),逐步发展为复杂壳(采用多层壳嵌套结构,且执行时每次脱壳例程仅还原部分原始代码)^[16]。相应地,攻击者的保护策略也从单一加壳工具简单保护,升级为多层壳嵌套与多种加壳工具协同使用的复杂方案^[17]。

加壳技术对恶意代码抗分析能力的增强,主要体现在对程序结构的破坏与执行信息的隐藏两个方面。一方面,样本的标准文件结构遭到破坏,导致反汇编操作难以有效开展,原始代码语义信息无法准确还原;另一方面,运行时的关键依赖信息(如包含 API 调用信息的导入表)常被加壳工具清除,这会给分析环境准备阶段的样本行为监控部署带来较大阻碍。

2.1.3 无文件攻击

无文件攻击是一种不依赖可执行文件开展恶意行为的新型攻击范式^[18],其典型攻击流程如图 2 所示。区别于传统攻击方式,此类攻击通过将恶意载荷直接注入内存运行,并深度利用目标系统的合法内置工具(如 PowerShell、Office 宏等)实施攻击行为,避免了在目标系统中植入持久化恶意软件。凭借其高隐蔽性和低可追溯性特征,无文件攻击已发展成为当前最具威胁的网络攻击手段之一^[19],给恶意代码动态分析技术带来了全方位挑战。

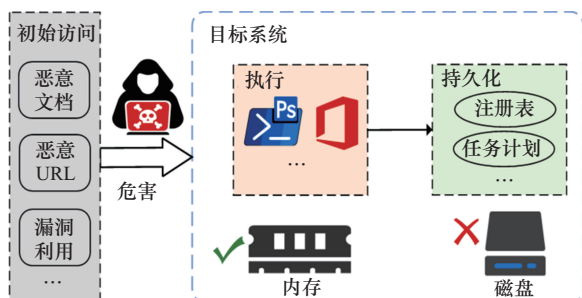


图 2 无文件攻击的典型攻击流程

无文件攻击的典型特征对动态分析流程的各个环节均构成显著影响。

首先,无文件攻击的恶意载荷以非磁盘文件的形式存在,规避了预处理阶段针对可执行文件实体的常规分析手段。这类攻击具有“不落地”(living-off-the-land, LotL)特点^[20],其生存维持依赖于在注册表等原生系统资源中驻留,而非在目标系统植入持久化恶意软件,这显著增加了恶意载荷的捕获与分析难度。由于无须在磁盘生成恶意文件,常规预处理阶段通过解析可执行文件获取特征的方法难以奏效。

其次,无文件攻击的动态分析高度依赖精准复现目标环境,对分析环境准备提出了严苛要求。无文件攻击通常通过漏洞利用链获取执行权限,其成功执行往往需要特定的环境条件。分析系统必须构建与真实受害环境高度一致的运行环境(包括操作系统版本、补丁状态、服务配置等),任何环境偏差都可能导致攻击行为无法触发。

最后,在样本执行与行为捕获阶段,聚焦进程行为的监控方式在无文件攻击场景中不再适用。无文件攻击通过合法系统工具(如 PowerShell 脚本、Office 应用程序等)实施恶意操作,要求分析系统必须采用多维度监控策略,对进程行为、内存操作、API 调用、网络通信等进行关联分析,才能有效捕获其恶意行为特征。

2.2 分析环境感知

分析环境感知技术旨在检测软件是否运行在分析环境中,并依据检测结果决定是否执行恶意行为。恶意代码常通过检测调试器特征、沙箱痕迹和用户使用情况感知分析环境。

2.2.1 调试器检测

调试器是一种重要的程序分析工具,它通过设置断点、监控和修改程序运行时的状态来实现程序调试。用户通过外部输入设备与调试器交互。

在恶意代码分析领域,调试器检测已成为对抗动态分析的关键技术。恶意代码通常通过感知调试状态与正常执行状态的差异实现调试器检测,具体手段如下。

1) 直接读取调试环境指纹。调试器加载样本程序时,会在系统中留下调试标志位等指纹信

息，恶意代码可通过系统接口直接查询这些调试状态信息。

2) 捕获调试操作的可观测痕迹。调试操作会引入可观测痕迹。在调试过程中，断点设置会在代码段插入断点标志，而程序因断点暂停产生的异常耗时也会形成时间偏差。恶意代码可以通过校验代码段完整性（如哈希比对）或时钟计时^[21]捕获这些痕迹，进而推断是否被调试。

3) 通过异常处理逻辑验证环境。调试器会接管程序的异常处理流程，这可能破坏恶意代码预设的异常处理逻辑。恶意代码通过主动触发异常并观察处理路径是否被劫持，即可间接判定调试器活动。

4) 扫描调试器关联的标识信息。调试器运行时生成特定窗口，其窗口类名、标题等特征可作为检测指标，恶意代码通过扫描系统窗口句柄等方式可直接定位调试器关联特征。

2.2.2 沙箱检测

沙箱是一种自动化动态分析工具，通常用于恶意软件的分析与检测。其核心机制是在隔离的环境中运行程序样本，通过系统 API Hook、动态二进制插桩（dynamic binary instrumentation, DBI）等技术，在 I/O 活动、系统调用等层次监测样本的行为，并生成分析报告。为实现环境状态的快速恢复，传统的沙箱通常基于模拟器或虚拟机搭建样本的运行环境。

恶意代码通过识别沙箱软件的运行环境痕迹和行为监控痕迹来实现沙箱检测。

1) 运行环境检测。为便于环境配置和状态恢复，传统的沙箱使用模拟器和虚拟机等工具搭建样本运行环境。与真实物理机系统相比，这类虚拟环境在硬件、软件、网络等层次存在特征差异。恶意代码通过检测这些环境特征，可识别自身是否运行在沙箱系统中^[22-25]。目前，多数用于沙箱检测的环境特征仍须通过人工逆向分析获取，但部分特征已可实现自动化提取。Shi 等^[26]以 Intel x86 指令手册为依据设计测试用例生成规则，通过在虚拟机与真机环境中自动化运行测试指令并对比执行结果差异，以发现可用于虚拟机环境检测的指令特征。

2) 行为监控检测。沙箱通常采用 API Hook、

DBI 侵入式监控技术捕获样本的 API 调用等行为数据，但这些技术会在内存中留下可被感知的痕迹。Chaillytko 等^[27]设计了栈平衡校验方法，用于检测 Cuckoo 沙箱的 API Hook 痕迹；D'Elia 等^[28]系统调研了 DBI 工具的攻击面，发现其存在可被恶意代码感知的透明性缺陷。

2.2.3 用户活动检测

恶意代码动态分析系统的运行环境与日常使用的真实物理机在用户活动痕迹方面存在显著差异，这种差异主要体现在两个维度：其一，在自动化分析流程中，用户与样本的交互行为极少；其二，为降低无关因素干扰，分析系统提供的运行环境通常较为“干净”，缺少真实设备长期使用后形成的“系统老化”特征。

针对用户交互活动少的特点，恶意代码通过“反向图灵测试”验证环境真实性。例如，在执行过程中随机弹出确认信息窗口等。针对运行环境缺少“系统老化”特征的特点，恶意代码通过扫描系统关键位置的“磨损痕迹”识别环境属性。相关研究表明，研究人员通过对比真实用户设备与分析环境在文件系统、配置信息等方面的差异，成功提取出可用于环境识别的“磨损”特征，并在 PC 和移动平台上实现了对分析环境的感知^[29-30]。

2.3 行为捕获逃避

为避免恶意行为被动态分析系统捕获，恶意代码采用时延和条件触发技术隐藏恶意行为，通过干扰分析过程对抗行为监控。

2.3.1 时延

在恶意代码动态分析中，自动化分析系统通常会设定分析时间上限，若样本执行时间超过预设阈值，分析流程会被强制终止。恶意代码正是利用这一特性，通过时延策略推迟恶意功能的执行时间，从而逃避动态分析的捕获。时延主要有睡眠函数调用和迟滞代码两种实现方式。睡眠函数调用，即通过调用 Sleep() 等系统 API 让程序睡眠指定时间，其实现方式简单直接，容易被分析系统通过 API 监控识别，从而暴露逃避意图。迟滞代码是指通过执行大量冗余操作消耗时间，间接实现延迟效果。与睡眠函数调用相比，这种方式的隐蔽性更强。

2.3.2 条件触发

条件触发是指恶意软件仅在目标环境满足预设条件时才执行恶意操作。尽管条件触发与分析环境感知技术均需要收集环境信息,但二者的核心目标存在显著差异。此类恶意软件更关注与业务场景相关的特征(如系统语言、当前日期、软件版本等),其主要目的并非识别是否处于分析环境,而是确保恶意载荷仅在预期的目标环境中被激活。例如,“震网”(Stuxnet)病毒便以目标系统是否安装西门子工业控制软件作为激活后续攻击流程的必要条件^[31]。这种仅在特定条件被满足后才执行的代码片段,通常被称为“逻辑炸弹”。

2.3.3 分析过程干扰

恶意代码通过规避调试跟踪和逃避行为监控等方式干扰分析过程,在确保恶意操作正常执行的同时有效规避行为监控系统的检测。

恶意代码可利用调试器的机制特点规避调试跟踪。例如,干扰调试器与调试子系统之间的隐式通信,可使样本创建不被调试器追踪的进程或线程^[32]。此外,通过锁定键盘或鼠标、强制切换桌面等方式干扰交互,也能有效破坏调试过程^[33]。

监控 API 调用序列是沙箱常用的行为监控方法。其常用的实现方式是利用 Hook 技术修改 API 函数的头部指令,将控制流重定向到样本监控模块的处理例程中。为避免恶意行为被捕获,恶意代码常采用 Sliding Call 和 Stolen Code 技术绕过 API 头部的 Hook 代码。此外,恶意代码可通过规避系统 DLL 的加载与调用机制隐藏 API 调用,如自主实现 DLL 装载的 Stealth Loader^[34]和基于地址空间模糊混淆调用关系的 APIASO^[35]方案。

2.4 分析模型欺骗

动态分析系统基于程序运行时产生的 API 调用序列、系统行为等数据完成分析任务。为逃避这类检测,恶意代码通过扰动序列数据和混淆行为轨迹的方式欺骗分析模型,以破坏行为数据分析结果的准确性。

2.4.1 序列数据对抗扰动

作为行为数据分析阶段的一种核心输入,

API 调用序列容易受到干扰,攻击者可通过对抗样本生成技术对模型判定实施系统性攻击。Rosenberg 等^[36]首次提出针对 API 调用序列分类器的端到端黑盒攻击框架 GADGET。该框架通过梯度引导的方式插入无害 API 来干扰模型的注意力。为降低攻击成本,其后续研究^[37]结合生成对抗网络与自适应进化算法,有效减少了查询次数。Chen 等^[38]提出基于注意力机制的 API 排序算法,优先在关键 API 位置插入扰动,并利用词移距离优化对抗样本的语义隐蔽性。

为确保对抗样本的隐蔽性与功能完整性,研究者开发了多种实用化生成技术。Kucuk 等^[39]通过插入虚假调用等源码级修改方式,将恶意代码样本定向误分类至特定家族。Digregorio 等^[40]结合位置敏感快速梯度符号方法生成最小扰动的 API 序列,并采用最长已知行为和级联行为优化策略应对动态执行的不确定性;同时通过直接修改二进制字节码保留样本功能,进一步提高对抗样本的隐蔽性。

2.4.2 恶意行为轨迹混淆

针对基于系统操作行为构建的系统行为图分析模型,攻击者可通过构造符合正常行为模式的操作序列,将恶意行为轨迹伪装成合法操作链,从而干扰检测模型对系统因果依赖关系的准确理解。在确保恶意逻辑功能完整性的前提下,攻击者会将恶意操作隐蔽地嵌入合法的行为路径中,以实现和分析模型的欺骗性攻击。

研究者通过多种技术路径验证了恶意行为轨迹混淆的有效性。Goyal 等^[41]的研究揭示了图编码机制存在结构性缺陷,他们通过注入良性子图的方式,降低恶意行为结构在检测中的显著性,最终在 5 种主流检测系统中成功实现逃逸。在此基础上,Mukherjee 等^[42]进一步细化了攻击粒度,提出基于事件发生频率的分析方法,识别出对检测模型影响较大的低频率显著事件,并生成语义等价的对抗性替代事件链,从而实现更为隐蔽的攻击。Sang 等^[43]则针对图分析系统的关键指标生成混淆目标。该方法通过建立图模型嵌入表示与真实系统操作之间的映射关系,在混淆目标的指引下生成可直接部署的混淆程序,显著降低了现有检测方法的准确率。

3 恶意代码动态分析反对抗技术

为增强动态分析系统的对抗性防御能力, 研究人员构建了恶意载荷捕获、环境感知规避检测、行为触发与监控增强及分析模型优化等维度的技术增强体系。

3.1 恶意载荷捕获

研究人员已设计出代码反混淆与软件脱壳方案, 可对经过混淆和加壳处理的恶意软件进行逆向分析; 同时, 研究人员从无文件攻击的载体检测、场景还原及行为分析3个维度开展研究, 提升分析系统对恶意载荷的捕获能力。

3.1.1 代码反混淆

代码反混淆是代码混淆的逆过程, 其在恶意代码分析中的核心用途是从经过混淆处理的样本中提取出有效的恶意载荷。根据对混淆工具特征的依赖程度, 反混淆技术可分为基于特征的反混淆和通用反混淆。

基于特征的反混淆通过识别或提取目标混淆工具的独有特征来指导反混淆过程。例如, 通过构造特定输入并观测混淆程序的执行结果, 可提取代码虚拟化混淆器的指令映射规则等信息, 从而实现反混淆^[44-45]。这类方法的优势在于针对性强、运行效率高; 其局限性在于难以应对未知或自定义的混淆技术, 一旦混淆工具升级或采用新的混淆策略, 基于特征的方法可能失效。

通用反混淆不依赖特定混淆器的特征, 而是采用通用的分析手段进行反混淆。在数据流反混淆方面, 针对结合布尔逻辑运算与算术运算的混合布尔算术表达式 (mixed Boolean-arithmetic, MBA) 混淆技术, 已有研究通过代数化简、程序合成和规则学习等手段, 实现对线性 and 非线性表达式的有效还原^[46-47]。在控制流反混淆领域, Mariano 等^[48]提出了一种执行轨迹引导的组合式程序合成反混淆方法, 该方法能够有效应对控制流平坦化等6种控制流混淆技术及其组合。

3.1.2 软件脱壳

软件脱壳是软件加壳的逆过程, 在恶意代码分析中的核心用途是从经过加壳处理的样本中提取有效的恶意载荷。主流脱壳技术可分为基于已知壳识别的针对性脱壳和通用脱壳两类。

针对性脱壳通过壳识别执行相应的脱壳操

作。基于熵值、节数量等特征训练的机器学习模型可实现加壳的自动化检测^[49]。此外, 规则匹配可高效识别壳, 但其依赖人工提取特征, 存在维护成本高、误报率高的问题。为此, 研究人员提出通过动态与静态分析选取壳的独有特征, 自动生成检测规则, 以降低误报率^[50]。

通用脱壳不依赖壳识别, 而是通过监控程序执行时的固有特征实现自动化脱壳。Cheng 等^[51]利用恶意代码脱壳时会重建导入表并通过其调用 API 这一典型特征, 定位原始入口点。在后续研究中, 他们采用 API 微执行技术和 Intel BTS 等硬件特性修复被隐藏的导入表, 提高脱壳成功率^[52]。针对脱壳过程中的密集解密操作会触发特定模式的硬件计数器变化这一特点, LoopH-PCs^[53]使用 Intel PEBS 和最后分支记录 (last branch record, LBR) 机制, 结合机器学习来准确识别原始代码层脱壳行为, 实现高透明度的自动化脱壳。

3.1.3 无文件攻击检测

为应对无文件攻击带来的挑战, 研究人员从攻击载体检测、攻击场景还原和攻击行为分析等维度, 开展对该类攻击的检测与分析工作, 提升分析系统的恶意载荷捕获能力、分析环境构建能力和行为监控能力。

攻击载体检测聚焦被无文件攻击滥用的 PowerShell 脚本、Office 宏文档和离地攻击二进制程序等工具的检测。研究人员采用基于抽象语法树的静态分析、动态执行追踪等方式实现对 PowerShell 脚本的反混淆^[54-56], 并对其进行恶意性检测和家族分类^[57-58]。针对在网络钓鱼中被滥用的宏文档, Ruaro 等^[59]通过符号执行技术实现对 Excel 4.0 宏的反混淆; 李晨光等^[60]构建深度学习模型实现对 Excel 4.0 宏的家族分类。在离地攻击检测方面, Ongun 等^[61]提出了首个 LotL 攻击的主动学习框架。该框架利用词嵌入对命令行文本进行表示, 并构建分类器来识别攻击命令。

攻击场景还原通常依赖网络威胁情报 (cyber threat intelligence, CTI) 来构建高保真的分析环境。在无文件攻击分析中, CTI 提供的漏洞描述、攻击者技战术等信息, 有助于提高分析环境的针对性与行为触发的有效性。例如, 公开的

漏洞描述能够协助构建漏洞复现环境^[62], 攻击分析报告经深度学习模型处理, 能够生成包含攻击常用漏洞、发起攻击环境等要素的攻击画像^[63]。CTI 的处理需结合应用场景, 借助自然语言处理等技术实现语义理解与实体关系抽取, 最终支撑威胁知识图谱的自动化构建。AttacKG 方法^[64]基于 ATT&CK 技术模板描述通用技术知识, 并利用图对齐算法实现攻击图合并, 拓展检测与分析的覆盖范围。IntelEX 方法^[65]利用大语言模型提取 CTI 报告中的攻击技战术信息, 以支持攻击模拟等任务。

攻击行为分析主要依托溯源图来还原攻击路径。溯源图是一个有向图, 其中节点代表进程、文件等实体, 边则表示进程的创建与结束、文件读写等实体间的关系, 图 3 展示了一个典型的溯源图。基于溯源图的攻击行为分析可划分为溯源图的构建、优化和分析 3 个阶段。溯源图构建通过标准化系统审计日志^[66]、利用硬件追踪技术恢复细粒度数据流^[67]、融合应用日志和系统日志多源数据^[68]等方法形成对系统行为的全面准确描述; 溯源图优化通过噪声过滤和路径因子剪枝^[69]、基于图蒸馏技术的溯源图压缩^[70]等方式, 解决状态爆炸问题; 攻击分析通过图概要与进化模型学习^[71]、基于图神经网络的节点级异常检测^[72]、社区发现算法生成攻击摘要^[73]等手段, 实现对攻击行为的识别和描述。

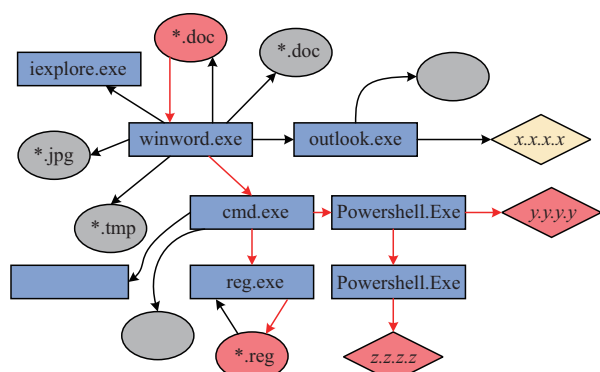


图 3 溯源图示例

3.2 环境感知规避

为规避恶意软件的环境感知行为, 研究人员通过检测环境感知代码、构建透明分析环境和多系统执行的方式对其进行检测和规避。

3.2.1 环境感知代码检测

类似于基于签名的恶意软件检测方案, 环境感知代码检测通过规则匹配来识别样本是否采用分析环境感知技术。例如, 大量使用的 `cpuid` 和 `rdtsc` 指令, 可作为基于时间开销的环境感知行为的特征^[23]。BluePill^[74]基于已知环境感知技术的特征, 在指令、API 和系统调用等层面开展对分析对抗行为的反制。此外, 基于特征的检测方案还被用于调研大规模恶意样本中的分析对抗行为^[75]。环境感知代码检测技术能够有效识别采用已知技术的恶意软件, 但难以应对新型对抗技术。

3.2.2 透明分析环境构建

透明分析环境构建是一类通用的环境感知规避技术。研究人员通过隐藏环境特征、利用硬件辅助虚拟化技术、基于真机构建分析系统等方式, 减少分析环境与真实用户环境之间的可感知差异。

表 1 对该方向的主要研究工作进行了总结。

1) 基于环境特征隐藏的透明分析环境构建。此方案的设计思路为拦截样本对环境信息的查询请求, 并修改其返回值以隐藏分析环境特征^[74]。其优点是无须对运行环境架构进行较大的改动, 缺点是难以应对未披露的分析环境感知技术。

2) 基于硬件辅助虚拟化的透明分析环境构建。相较于软件模拟系统, 基于硬件辅助虚拟化的虚拟机系统性能更强、环境特征暴露较少。此类分析系统通常在客户机中运行样本, 通过虚拟机管理 (virtual machine manager, VMM) 软件提供的指令流监控、内存访问审计和系统调用拦截等机制, 实现对样本执行路径的自动化监控。在此基础上, 借助客户机与 VMM 运行模式切换、扩展页表等机制, 还能够实现单步执行与断点设置, 支持对样本程序的调试分析^[76-79]。此类分析系统的运行环境状态恢复较为方便, 但难以消除指令运行偏差等可用于环境感知的虚拟机系统特征。

3) 基于真机的透明分析环境构建。基于真机系统 (Bare-metal) 构建的分析环境具有良好的透明度。此类研究或采用类似于取证分析的粗粒度行为画像技术, 观测样本的文件读写、注册表

项修改等对运行环境的影响信息^[80-81]；或配合设备硬件特性实现细粒度行为监控，如利用CPU分支记录特性捕获样本的API调用序列^[82]。此外，基于真机的调试器实现通常需要更高特权等级的设备硬件特性的配合^[82-83]。此类分析环境透明度高，但其实现与部署较复杂。

从表1可以看出，动态分析中的行为监控方式由运行环境的构建方法决定，且实现复杂度会随运行环境透明度的提高而增加。图4揭示了这一演变过程的深层机理，即动态分析系统的演进本质上是攻防双方持续博弈的结果。

动态分析系统的运行环境已完成从模拟器、虚拟机到真机的演化。攻击者通过识别分析环境与用户真实环境之间的差异来检测分析系统；防御方则通过不断改进运行环境构建方法来提高其透明度。然而，攻防双方存在天然的博弈不对称性，运行环境中任何微小的痕迹都可能暴露分析意图，从而危及系统的隐蔽性。因此，动态分析系统的运行环境正朝着无限逼近真实用户系统的方向持续优化。

动态分析系统的行为监控方式始终与其运行环境构建技术的发展保持同步演进，呈现出显著的技术适配特征。在软件模拟运行环境下，分析

系统主要采用侵入式行为监控方式（如API Hook）实现行为捕获；在硬件辅助虚拟化运行环境下，分析系统普遍依托VMM进行行为捕获；在真机运行环境下，分析系统则通过设备硬件特性实现细粒度行为的观测与捕获。这种演进路径清晰地表明：通过获取更高特权等级的接口来实现行为监控，已成为动态分析系统行为捕获的核心技术范式。

值得注意的是，当前工业界的动态分析系统仍以虚拟化平台为主。虚拟机系统的安全隔离、资源调度及自动化管理等方面优势显著，能够高效支撑大规模样本分析；而透明度更高的真机系统受成本因素限制，通常仅用于高价值样本的专项分析。

3.2.3 多系统执行

恶意软件采用的分析环境感知技术已成为动态分析面临的重要挑战。当检测到自身处于分析系统时，这类恶意软件会主动隐藏后续恶意行为，导致其在分析环境与真实用户环境中表现出显著的行为差异。针对这一逃避特性，研究人员在提高环境透明度的基础上，创新性地提出了基于多系统执行差异的检测方法。该方法的核心思想是通过对比样本在不同环境中的行为特征差

表1 透明分析环境构建工作总结

类别	运行环境	行为监控实现方式	相关文献	优点	缺点
基于环境特征隐藏的透明分析环境构建	虚拟机或真机	基于DBI技术实现行为分析	[74]	无须调整运行环境架构	难以应对未披露的分析环境感知技术
基于硬件辅助虚拟化的透明分析环境构建	虚拟机	基于VMM提供的接口，获取客户机信息，从而监控客户机中运行的样本程序行为	[76-77] [78-79]	运行环境状态恢复方便	存在难以消除的环境特征
基于真机的透明分析环境构建	真机	观测文件读写等对运行环境的影响信息 使用设备硬件特性实现细粒度行为监控	[80-81] [82-83]	环境透明度高	实现与部署较复杂

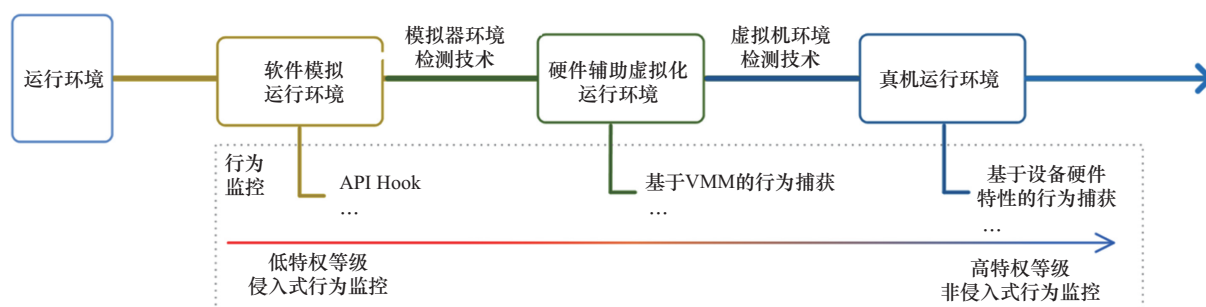


图4 动态分析系统的演化过程

异, 识别具有环境感知能力的规避型恶意软件。表 2 汇总了基于多系统执行差异的规避型恶意软件检测研究工作。这些研究方案均可检测环境敏感的恶意软件, 其中 MalGene^[84]采用细粒度系统调用序列作为分析特征, 不仅能有效检测环境感知型恶意软件, 还可进一步提取样本的逃避行为特征。基于多系统执行差异的规避型恶意软件检测不针对特定的环境感知技术, 是一种通用检测方案, 其有效性取决于运行环境的设置, 尤其是作为对照的运行环境的透明度。

3.3 行为触发与监控增强

为应对恶意代码的行为捕获逃避技术, 研究人员主要从两个方向展开工作: 一方面通过优化动态分析方法, 采用跳过时延、探索样本执行路径等技术手段, 有效触发样本的隐藏恶意行为; 另一方面着力提升行为监控模块的抗干扰能力, 确保对恶意代码行为特征的全面捕获。

3.3.1 时延跳过

针对恶意代码中通过调用 Sleep() 等 API 实现的显式时延, 分析系统可采用 API Hook 等技术手段直接跳过延迟代码的执行。针对迟滞代码等隐式时延, HASTEN 方法^[87]利用启发式规则, 通过监控系统调用触发情况判断程序是否陷入迟滞状态, 其采用减少对迟滞循环位置的监控、主动操控控制流以提前终止迟滞代码的执行等策略来跳过延迟。此外, K  chler 等^[88]的研究表明, 时延对恶意代码检测效率的影响有限, 在大多数分析场景下, 设置 2 min 的执行超时阈值即可有效捕获恶意行为。

3.3.2 执行路径探索

为全面触发恶意样本的潜在行为, 动态分析系统必须覆盖尽可能多的执行路径, 以揭示其隐藏的恶意逻辑。由于恶意代码通常依赖特定触发条件, 研究人员主要采用外部输入调控和强制执

行两类技术手段来探索不同的执行路径。

(1) 基于外部输入调控的路径探索

样本的执行路径高度依赖运行环境配置及外部输入。针对这一特性, 研究者提出了一系列优化方案。GoldenEye 方法^[89]采用渐进式预执行策略结合污点分析技术, 推测基本块执行后的分支情况, 无须完整执行所有路径即可高效触发隐藏行为, 显著减少了分析开销。Alaeiyan 等^[90]设计了虚拟环境条件生成器, 通过多次执行并分析日志, 识别未满足的触发条件, 并在后续执行中动态提供所需资源, 从而激活恶意代码的隐蔽逻辑。在此基础上, 该团队在 Sober^[91]中加入了用户行为模拟模块, 进一步增强了针对交互式恶意代码的触发能力。

(2) 基于强制执行的路径探索

强制执行技术通过直接干预程序控制流, 无须依赖环境或输入调整即可探索不同路径。X-Force^[92]是强制执行的代表性方案, 其核心思路是应用 DBI 技术强制改写与执行路径相关的分支条件指令, 直接操纵执行流。在 APT 恶意代码样本测试中, X-Force 成功触发更多的隐藏恶意行为。PMP 系统^[93]通过在程序执行前预分配内存, 并用随机值初始化, 规避无效指针解引用引发的异常, 同时减少状态冲突, 在减少运行开销的同时提升路径覆盖率。针对传统线性搜索方法因忽略数据流依赖关系而导致路径探索效率低下(存在大量冗余路径)和恶意行为暴露不完整的问题, He^[94]等提出了 DueForce 解决方案。该方案引入了动态数据流分析技术, 通过追踪“定义—使用”关系指导路径选择, 有效减少了冗余路径探索, 并提高了关键恶意行为的触发率。

3.3.3 监控模块加固

动态分析系统常采取针对性技术手段加固监控模块, 以应对已知的分析过程干扰技术。为增

表 2 基于多系统执行差异的规避型恶意软件检测研究工作总结

研究工作	功能	分析输入	运行环境设置
BareCloud ^[80]	规避型恶意软件检测	行为画像	Anubis、Ether ^[76] 和 Cuckoo 沙箱作为分析环境, 真机系统作为对照环境
MalGene ^[84]	规避型恶意软件检测和样本逃避特征提取	系统调用序列	Anubis 作为分析环境, Ether 作为对照环境
文献 ^[85]	规避型恶意软件检测	系统调用序列	Anubis (模拟器) 的改版和 VMware 虚拟机, 互为对照
DISARM ^[86]	规避型恶意软件检测	行为画像	基于 Anubis 构建的 4 个不同配置的沙箱系统, 互为对照

强调试器应对反调试技术的能力, 可通过 API Hook 等方式拦截对抗技术的控制流。针对 API 混淆问题, API-Xray^[52]利用 Intel BTS 技术追踪样本控制流, 并将长跳转目的地址与系统 API 所在的内存范围进行匹配, 替代传统基于 API 入口地址匹配的监控方式, 以捕获混淆后的 API 调用, 从根本上应对 Stolen Code 等混淆技术。

3.4 分析模型增强

为应对分析模型欺骗技术, 研究人员从模型鲁棒性提升与检测规则动态优化入手, 系统性增强恶意代码分析模型应对复杂攻击的能力。

3.4.1 模型鲁棒性提升

传统检测模型由于过度依赖局部行为特征, 在面对序列扰动、行为混淆等攻击手段时, 极易出现检测偏差或误判。为解决这一问题, 研究人员提出了基于行为单元开展学习和将根源分析嵌入图神经网络两类方法来提升检测模型的鲁棒性。

在行为序列分析领域, Zhan 等^[95]通过提取具有明确功能意图的行为单元, 将局部行为的语义完整性作为抗干扰基础, 结合多级深度学习模型学习单元内部语义与单元间上下文关系, 从而在对抗样本攻击下仍能保持稳定的检测精度。

在基于溯源图的攻击检测领域, R-CAID 系统^[96]采用预计算节点根本原因的方法, 使检测模型能够准确识别攻击链的初始入口特征 (如恶意进程的启动源)。即使攻击者尝试伪造行为上下文, 系统仍能通过捕捉初始访问阶段留下的不可变特征来影响分类决策, 从而有效抵御基于局部行为模仿的逃避攻击策略。

3.4.2 检测规则动态优化

基于规则的检测方法虽然高效, 但容易受到变种攻击的规避。为应对这一挑战, 研究人员提出了动态优化检测规则方案, 通过持续更新规则库和增强分析模型来提升系统对抗逃避攻击的能力。APIARY 框架^[97]通过深度分析软件行为, 提取关键 API 调用特征, 并利用良性与恶意软件在 API 调用模式上的显著差异, 自动生成高效的检测规则。实验表明, 该框架生成的规则在多个基准数据集上的检测性能优于传统方法。AMIDES 系统^[98]专注于解决安全信息与事件管理 (security information and event management, SIEM) 规则逃避

问题。该系统通过机器学习对比输入事件与 SIEM 规则及历史良性事件的相似性, 识别潜在逃避行为, 并通过规则归因分析辅助定位被绕过的规则。

4 总结与展望

动态分析技术已成为恶意代码检测与分析的核心手段。随着攻防对抗的持续升级, 分析环境的透明化程度不断提高, 传统对抗策略在人工智能等新兴技术的推动下正加速演进。特别是无文件攻击等新型威胁的出现, 不仅推动分析模型从局部行为监控向系统级状态追踪转变, 还强化了威胁情报在检测闭环中的支撑作用。基于上述分析, 本文对该领域的未来研究方向提出以下展望。

1) 基于虚拟化平台的新型动态分析系统对抗演化。虚拟化平台作为大规模样本分析的主流架构, 其环境透明性与运行效能的持续优化引发了新一轮攻防博弈。攻击者不断开发新型虚拟化环境感知技术以提升隐蔽性, 而防御方则通过特征隐藏、调度优化等手段构建更透明、高效的分析环境。未来, 双方将围绕虚拟化环境构建、行为监控机制与反检测能力展开博弈, 推动动态分析系统在架构设计与运行机制上的迭代升级。

2) 底层硬件事件与上层软件行为之间的语义映射研究。构建透明分析环境的典型技术路径在于利用高特权级接口实现对样本行为的观测。当前, 基于真机环境构建的分析系统具有最优的透明度, 而运行在更高特权等级的 CPU 等硬件设备提供的功能接口, 为实现细粒度行为观测提供了支撑。例如, CPU 的分支记录追踪功能已被用于提取样本的 API 调用序列等执行路径信息^[52,82], 硬件性能计数器也被用于识别恶意软件脱壳过程中出现的“热循环”特征^[53]。未来, 通过解决底层硬件事件与上层程序行为之间的“语义鸿沟”, 建立二者间的对应关系, 能够将行为观测推广至迟滞代码识别等行为分析场景, 实现针对恶意代码的特异性行为分析。

3) 结合大语言模型的恶意代码动态分析对抗与反对抗技术实现。恶意代码分析技术已历经多个发展阶段: 从早期依赖专家经验的规则匹配, 到基于沙箱和机器学习模型的自动化判定, 再到当前网络攻击的智能化溯源与推理^[99]。在这一演

进过程中,以 ChatGPT、DeepSeek 等为代表的大语言模型凭借其卓越的上下文理解能力、多步推理能力及任务适应能力,为恶意代码分析领域注入了新的智能化动力。目前,这些大语言模型已在代码混淆^[15]、CTI 解析^[65]等攻防场景中得到实际应用。在动态分析攻防对抗的传统研究领域(如分析环境感知、样本执行路径探索等),具体实现过程中的关键决策环节往往依赖于启发式规则,其智能化水平仍有较大提升空间。因此,如何深入挖掘大语言模型在恶意代码动态分析对抗与反对抗中的应用潜力,特别是在复杂环境感知、样本执行路径智能化探索等关键任务中的适用性,将成为未来值得重点关注的研究方向。

4) 威胁情报驱动的动态分析环境配置。在恶意代码分析实践中,精准复现已披露样本的攻击行为是一项关键任务。然而,当前分析环境的配置主要依赖人工经验或碎片化的威胁情报整合,缺乏系统化、自动化的智能配置方案。近年来, VirusTotal 等平台提供的多引擎分析报告,以及 MITRE ATT&CK、Engage 等网络攻防知识体系对攻击行为的结构化描述,为分析环境要素的智能选取与自动化配置提供了重要参考。未来,基于多源威胁情报自动化提取环境依赖特征,驱动分析环境的个性化定制,以精准复现恶意软件攻击链,将成为提升分析效率与准确性的关键研究方向。

5 结束语

动态分析是恶意代码分析与检测的关键技术。攻防双方围绕动态分析的各个环节展开了激烈博弈。本文以恶意代码动态分析的工作流程为主线,系统梳理了动态分析对抗与反对抗技术,揭示了分析系统迭代升级的原因。此外,本文对该领域未来研究方向进行了展望,旨在为构建更加透明、智能的恶意代码动态分析系统提供参考。

参考文献:

- [1] 工业和信息化部电子第五研究所. 2023 上半年网络安全观察报告[R]. 2023.
The Fifth Research Institute of Electronics of the Ministry of Industry and Information Technology. 2023 first-half cybersecurity observation report[R]. 2023.
- [2] NSFOCUS. NSFOCUS reveals new botnet family RDDoS[EB]. 2024.
- [3] Krishnan R. Red CryptoApp: a new threat group in the ransomware world[EB]. 2024.
- [4] 360 数字安全. 2024 年全球高级持续性威胁(APT)研究报告[R]. 2025.
360 Digital Security. 2024 Global Advanced Persistent Threat (APT) Research Report[R]. 2025.
- [5] OR-MEIR O, NISSIM N, ELOVICI Y, et al. Dynamic malware analysis in the modern era-a state of the art survey[J]. ACM Computing Surveys, 2019, 52(5): 1-48.
- [6] GABER M G, AHMED M, JANICKE H. Malware detection with artificial intelligence: a systematic literature review[J]. ACM Computing Surveys, 2024, 56(6): 1-33.
- [7] AFIANIAN A, NIKSEFAT S, SADEGHIYAN B, et al. Malware dynamic analysis evasion techniques: a survey[J]. ACM Computing Surveys, 2020, 52(6): 1-28.
- [8] GENG J X, WANG J F, FANG Z Y, et al. A survey of strategy-driven evasion methods for PE malware: transformation, concealment, and attack[J]. Computers & Security, 2024, 137: 103595.
- [9] 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. Beijing: Beijing University of Posts and Telecommunications, 2015.
- [10] HOSSEINZADEH S, RAUTI S, LAURÉN S, et al. Diversification and obfuscation techniques for software security: a systematic literature review[J]. Information and Software Technology, 2018, 104: 72-93.
- [11] 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.
- [12] SHA Z H, SHU H, XIONG X B, et al. Model of execution trace obfuscation between threads[J]. IEEE Transactions on Dependable and Secure Computing, 2022, 19(6): 4156-4171.
- [13] SCHLOEGEL M, BLAZYTKO T, CONTAG M, et al. Loki: hardening code obfuscation against automated attacks[C]//Proceedings of the 31st USENIX Security Symposium (USENIX Security). Berkeley: USENIX Association, 2022: 3055-3073.
- [14] ZHANG Z, XUE J F, BAKER T, et al. COVER: enhancing virtualization obfuscation through dynamic scheduling using flash controller-based secure module[J]. Computers & Security, 2024, 146: 104038.
- [15] MOHSENI S, MOHAMMADI S, TILWANI D, et al. Can LLMs obfuscate code? A systematic analysis of large language models into assembly code obfuscation[C]//Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). Washington : AAAI, 2025, 39(23): 24893-24901.
- [16] UGARTE-PEDRERO X, BALZAROTTI D, SANTOS I, et al. SoK: deep packer inspection: a longitudinal study of the complexity of run-time packers[C]//Proceedings of the 2015 IEEE Symposium on Security and Privacy. Piscataway: IEEE Press, 2015: 659-673.
- [17] MURALIDHARAN T, COHEN A, GERSON N, et al. File packing from the malware perspective: techniques, analysis approaches,

- and directions for enhancements[J]. *ACM Computing Surveys*, 2022, 55(5): 1-45.
- [18] LIU S D, PENG G J, ZENG H T, et al. A survey on the evolution of fileless attacks and detection techniques[J]. *Computers & Security*, 2024, 137: 103653.
- [19] CrowdStrike. CrowdStrike 2024 global threat report[R]. 2024.
- [20] BARR-SMITH F, UGARTE-PEDRERO X, GRAZIANO M, et al. Survivalism: systematic analysis of windows malware living-off-the-land[C]//*Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP)*. Piscataway: IEEE Press, 2021: 1557-1574.
- [21] OYAMA Y. How does malware use RDTSC? a study on operations executed by malware with CPU cycle measurement[M]//*Detection of Intrusions and Malware, and Vulnerability Assessment*. Cham: Springer International Publishing, 2019: 197-218.
- [22] RAFFETSEDER T, KRUEGEL C, KIRDA E. Detecting system emulators[C]//*Proceedings of the 10th International Conference (ISC)*. Berlin: Springer, 2007: 1-18.
- [23] BRENGEL M, BACKES M, ROSSOW C. Detecting hardware-assisted virtualization[C]//*Proceedings of the 13th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*. Berlin: Springer, 2016: 207-227.
- [24] YOKOYAMA A, ISHII K, TANABE R, et al. SandPrint: fingerprinting malware sandboxes to provide intelligence for sandbox evasion[C]//*Proceedings of the 19th International Symposium on Research in Attacks, Intrusions, and Defenses (RAID)*. Berlin: Springer, 2016: 165-187.
- [25] ZHANG Z, CHENG Y Q, GAO Y S, et al. Detecting hardware-assisted virtualization with inconspicuous features[J]. *IEEE Transactions on Information Forensics and Security*, 2020, 16: 16-27.
- [26] SHI H, ALWABEL A, MIRKOVIC J. Car dinal pill testing of system virtual machines[C]//*Proceedings of the 23rd USENIX Security Symposium (USENIX Security)*. Berkeley: USENIX Association, 2014: 271-285.
- [27] CHAILYTKO A, SKURATOVICH S. Defeating sandbox evasion: how to increase the successful emulation rate in your virtual environment[R]. 2017.
- [28] D'ELIA D C, COPPA E, NICCHI S, et al. SoK: using dynamic binary instrumentation for security (and how you may get caught red handed) [C]//*Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*. New York: ACM, 2019: 15-27.
- [29] MIRAMIRKHANI N, APPINI M P, NIKIFORAKIS N, et al. Spotless sandboxes: evading malware analysis systems using wear-and-tear artifacts[C]//*Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP)*. Piscataway: IEEE Press, 2017: 1009-1024.
- [30] KONDRACKI B, AZAD B A, MIRAMIRKHANI N, et al. The droid is in the details: environment-aware evasion of Android sandboxes[C]//*Proceedings of 2022 Network and Distributed System Security Symposium*. Virginia: the Internet Society, 2022.
- [31] FALLIERE N, MURCHU L O, CHIEN E. W32.Stuxnet dossier[R]. Symantec Security Response, 2011.
- [32] YASON M. The art of unpacking[R]. 2007.
- [33] KONG W, YANG K. Unlocking lockscreen[EB]. 2013.
- [34] KAWAKOYA Y, SHIOJI E, OTSUKI Y, et al. Stealth loader: trace-free program loading for API obfuscation[C]//*Proceedings of the 20th International Symposium on Research in Attacks, Intrusions, and Defenses (RAID)*. Berlin: Springer, 2017: 217-237.
- [35] LI Y, KANG F, SHU H, et al. APIASO: a novel API call obfuscation technique based on address space obscurity[J]. *Applied Sciences*, 2023, 13(16): 9056.
- [36] ROSENBERG I, SHABTAI A, ROKACH L, et al. Generic black-box end-to-end attack against state of the art API call based malware classifiers[C]//*Proceedings of the 21st International Symposium on Research in Attacks, Intrusions, and Defenses (RAID)*. Berlin: Springer, 2018: 490-510.
- [37] ROSENBERG I, SHABTAI A, ELOVICI Y, et al. Query-efficient black-box attack against sequence-based malware classifiers[C]//*Proceedings of the Annual Computer Security Applications Conference*. New York: ACM, 2020: 611-626.
- [38] CHEN X H, CUI L, WEN H, et al. MalAder: decision-based black-box attack against API sequence based malware detectors[C]//*Proceedings of the 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. Piscataway: IEEE Press, 2023: 165-178.
- [39] KUCUK Y, YAN G H. Deceiving portable executable malware classifiers into targeted misclassification with practical adversarial examples[C]//*Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*. New York: ACM, 2020: 341-352.
- [40] DIGREGORIO G, MACCARRONE S, D'ONGHIA M, et al. Tarallo: evading behavioral malware detectors in the problem space[C]//*Proceedings of the 21st International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*. Berlin: Springer, 2024: 128-149.
- [41] GOYAL A, HAN X Y, WANG G, et al. Sometimes, you aren't what you do: mimicry attacks against provenance graph host intrusion detection systems[C]//*Proceedings of the 2023 Network and Distributed System Security Symposium*. Virginia: the Internet Society, 2023.
- [42] MUKHERJEE K, WIEDEMEIER J, WANG T, et al. Evading provenance-based ML detectors with adversarial system actions[C]//*Proceedings of the 32nd USENIX Security Symposium (USENIX Security)*. Berkeley: USENIX Association, 2023: 1199-1216.
- [43] SANG A Y, WANG Y C, YANG L, et al. Obfuscating provenance-based forensic investigations with mapping system meta-behavior[C]//*Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses*. New York: ACM, 2024: 248-262.
- [44] LI S J, JIA C F, QIU P D, et al. Chosen-instruction attack against commercial code virtualization obfuscators[C]//*Proceedings of the 2022 Network and Distributed System Security Symposium*. Virginia: the Internet Society, 2022.
- [45] LUO C K, MING J, FU J M, et al. Reverse engineering of obfuscated lua bytecode via interpreter semantics testing[J]. *IEEE Transactions on Information Forensics and Security*, 2023, 18: 3891-3905.
- [46] LIU B B, SHEN J F, MING J, et al. MBA-Blast: unveiling and simplifying mixed boolean-arithmetic obfuscation[C]//*Proceedings of*

- the 30th USENIX Security Symposium (USENIX Security 2021). Berkeley: USENIX Association, 2021: 1701-1718.
- [47] LEE J, LEE W. Simplifying mixed Boolean-arithmetic obfuscation by program synthesis and term rewriting[C]//Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM, 2023: 2351-2365.
- [48] MARIANO B, WANG Z T, PAILOOR S, et al. Control-flow deobfuscation using trace-informed compositional program synthesis [C]//Proceedings of the ACM SIGPLAN International Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA 2024). New York: ACM Press, 2024: 2211-2241.
- [49] BERTRAND-VAN-OUYSEL C H, DAM K H T, LEGAY A. Analysis of machine learning approaches to packing detection[J]. Computers & Security, 2024, 136: 103536.
- [50] LI S J, MING J, QIU P D, et al. PackGenome: automatically generating robust YARA rules for accurate malware packer detection[C]//Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM, 2023: 3078-3092.
- [51] CHENG B L, MING J, FU J M, et al. Towards paving the way for large-scale windows malware analysis: generic binary unpacking with orders-of-magnitude performance boost[C]//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM, 2018: 395-411.
- [52] CHENG B, MING J, LEAL E A, et al. Obfuscation-resilient executable payload extraction from packed malware[C]//Proceedings of the 30th USENIX Security Symposium (USENIX Security). Berkeley: USENIX Association, 2021: 3451-3468.
- [53] CHENG B, LEAL E A, ZHANG H, et al. On the feasibility of malware unpacking via hardware-assisted loop profiling[C]//Proceedings of the 32nd USENIX Security Symposium (USENIX Security). Berkeley: USENIX Association, 2023: 7481-7498.
- [54] LI Z Y, CHEN Q A, XIONG C L, et al. Effective and light-weight deobfuscation and semantic-aware attack detection for PowerShell scripts[C]//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM, 2019: 1831-1847.
- [55] CHAI H J, YING L Y, DUAN H X, et al. Invoke-deobfuscation: AST-based and semantics-preserving deobfuscation for PowerShell scripts[C]//Proceedings of the 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). Piscataway: IEEE Press, 2022: 295-306.
- [56] LI R J, ZHANG C Y, CHAI H J, et al. PowerPeeler: a precise and general dynamic deobfuscation method for PowerShell scripts[C]//Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM, 2024: 4539-4553.
- [57] YANG X Z, PENG G J, ZHANG D N, et al. PowerDetector: malicious PowerShell script family classification based on multi-modal semantic fusion and deep learning[J]. China Communications, 2023, 20(11): 202-224.
- [58] ZHANG S F, LI S Z, LU J C, et al. Power-ASTNN: a deobfuscation and AST neural network enabled effective detection method for malicious PowerShell Scripts[J]. Computers & Security, 2025, 154: 104441.
- [59] RUARO N, PAGANI F, ORTOLANI S, et al. SYMBEXCEL: automated analysis and understanding of malicious excel 4.0 macros[C]//Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE Press, 2022: 1066-1081.
- [60] 李晨光, 杨秀璋, 彭国军. Excel 4.0 宏自动化反混淆与家族分类系统[J]. 网络与信息安全学报, 2024, 10(3): 66-80.
- LI C G, YANG X Z, PENG G J. Automated deobfuscation and family classification system for Excel 4.0 macros[J]. Chinese Journal of Network and Information Security, 2024, 10(3): 66-80.
- [61] ONGUN T, STOKES J W, OR J B, et al. Living-off-the-land command detection using active learning[C]//Proceedings of the 24th International Symposium on Research in Attacks, Intrusions and Defenses. New York: ACM, 2021: 442-455.
- [62] RUAN B N, LIU J H, ZHANG C Q, et al. KernJC: automated vulnerable environment generation for Linux kernel vulnerabilities[C]//Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses. New York: ACM, 2024: 384-402.
- [63] 杨秀璋, 彭国军, 李子川, 等. 基于 Bert 和 BiLSTM-CRF 的 APT 攻击实体识别及对齐研究[J]. 通信学报, 2022, 43(6): 58-70.
- YANG X Z, PENG G J, LI Z C, et al. Research on entity recognition and alignment of APT attack based on Bert and BiLSTM-CRF[J]. Journal on Communications, 2022, 43(6): 58-70.
- [64] LI Z Y, ZENG J, CHEN Y, et al. AttackKG: constructing technique knowledge graph from cyber threat intelligence reports[C]//Proceedings of the 27th European Symposium on Research in Computer Security (ESORICS 2022). Berlin: Springer, 2022: 589-609.
- [65] XU M, WANG H, LIU J, et al. IntelEX: a LLM-driven attack-level threat intelligence extraction framework[J]. arXiv preprint, arXiv: 2412.10872, 2024.
- [66] HOSSAIN M N, MILAJERDI S M, WANG J A, et al. SLEUTH: real-time attack scenario reconstruction from COTS audit data[C]//Proceedings of the 26th USENIX Security Symposium (USENIX Security 2017). Berkeley: USENIX Association, 2017: 487-504.
- [67] ZENG J, ZHANG C Q, LIANG Z K. PalanTir: optimizing attack provenance with hardware-enhanced system observability[C]//Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM, 2022: 3135-3149.
- [68] SONG Y B, WANG K H, SUN X, et al. A multi-source log semantic analysis-based attack investigation approach[J]. Computers & Security, 2025, 150: 104303.
- [69] MILAJERDI S M, GJOMEMO R, ESHETE B, et al. HOLMES: real-time APT detection through correlation of suspicious information flows[C]//Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE Press, 2019: 1137-1152.
- [70] RABOOKI S F, LI B, FEBRINANTO F G, et al. GraphDART: graph distillation for efficient advanced persistent threat detection[J]. arXiv preprint, arXiv: 2501.02796, 2025.
- [71] HAN X Y, PASQUIER T, BATES A, et al. Unicorn: runtime provenance-based detector for advanced persistent threats[C]//Proceedings of the 2020 Network and Distributed System Security Symposium. Virginia: the Internet Society, 2020.
- [72] WANG S, WANG Z L, ZHOU T, et al. THREATTRACE: detecting

- and tracing host-based threats in node level through provenance graph learning[J]. *IEEE Transactions on Information Forensics and Security*, 2022, 17: 3972-3987.
- [73] CHENG Z J, LV Q J, LIANG J Y, et al. Kairos: practical intrusion detection and investigation using whole-system provenance[C]// *Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP)*. Piscataway: IEEE Press, 2024: 3533-3551.
- [74] D'ELIA D C, COPPA E, PALMARO F, et al. On the dissection of evasive malware[J]. *IEEE Transactions on Information Forensics and Security*, 2020, 15: 2750-2765.
- [75] GALLORO N, POLINO M, CARMINATI M, et al. A systematical and longitudinal study of evasive behaviors in windows malware[J]. *Computers & Security*, 2022, 113: 102550.
- [76] DINABURG A, ROYAL P, SHARIF M, et al. Ether: malware analysis via hardware virtualization extensions[C]// *Proceedings of the 15th ACM Conference on Computer and Communications Security*. New York: ACM, 2008: 51-62.
- [77] NGUYEN A M, SCHEAR N, JUNG H, et al. MAVMM: lightweight and purpose built VMM for malware analysis[C]// *Proceedings of the 2009 Annual Computer Security Applications Conference*. Piscataway: IEEE Press, 2009: 441-450.
- [78] DENG Z, ZHANG X Y, XU D Y. SPIDER: stealthy binary program instrumentation and debugging via hardware virtualization[C]// *Proceedings of the 29th Annual Computer Security Applications Conference*. New York: ACM, 2013: 289-298.
- [79] KARVANDI M S, GHOLAMREZAEI M, KHALAJ MONFARED S, et al. HyperDbg: reinventing hardware-assisted debugging[C]// *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. New York: ACM, 2022: 1709-1723.
- [80] KIRAT D, VIGNA G, KRUEGEL C. BareCloud: bare-metal analysis-based evasive malware detection[C]// *Proceedings of the 23rd USENIX Security Symposium (USENIX Security)*. Berkeley: USENIX Association, 2014: 287-301.
- [81] SPENSKY C, HU H Y, LEACH K. LO-PHI: low-observable physical host instrumentation for malware analysis[C]// *Proceedings of the 2016 Network and Distributed System Security Symposium*. Virginia: the Internet Society, 2016.
- [82] BOTACIN M, DE-GEUS P L, GRÉGIO A. Enhancing branch monitoring for security purposes: from control flow integrity to malware analysis and debugging[J]. *ACM Transactions on Privacy and Security*, 2018, 21(1): 1-30.
- [83] ZHANG F W, LEACH K, STAVROU A, et al. Using hardware features for increased debugging transparency[C]// *Proceedings of the 2015 IEEE Symposium on Security and Privacy*. Piscataway: IEEE Press, 2015: 55-69.
- [84] KIRAT D, VIGNA G. MalGene: automatic extraction of malware analysis evasion signature[C]// *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. New York: ACM, 2015: 769-780.
- [85] BALZAROTTI D, COVA M, KARLBERGER C, et al. Efficient detection of split personalities in malware[C]// *Proceedings of the 17th Network and Distributed System Security Symposium (NDSS 2010)*. Virginia: the Internet Society, 2010.
- [86] LINDORFER M, KOLBITSCH C, MILANI COMPARETTI P. Detecting environment-sensitive malware[C]// *Proceedings of the 14th International Symposium on Recent Advances in Intrusion Detection (RAID 2011)*. Berlin: Springer, 2011: 338-357.
- [87] KOLBITSCH C, KIRDA E, KRUEGEL C. The power of procrastination: detection and mitigation of execution-stalling malicious code[C]// *Proceedings of the 18th ACM Conference on Computer and Communications Security*. New York: ACM, 2011: 285-296.
- [88] KÜCHLER A, MANTOVANI A, HAN Y F, et al. Does every second count? time-based evolution of malware behavior in sandboxes[C]// *Proceedings of the 2021 Network and Distributed System Security Symposium*. Virginia: the Internet Society, 2021.
- [89] XU Z Y, ZHANG J L, GU G F, et al. GoldenEye: efficiently and effectively unveiling malware's targeted environment[C]// *Proceedings of the 17th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*. Berlin: Springer, 2014: 22-45.
- [90] ALAEIYAN M, PARSA S, CONTI M. Analysis and classification of context-based malware behavior[J]. *Computer Communications*, 2019, 136: 76-90.
- [91] ALAEIYAN M, PARSA S, VINOD P. Sober: explores for invasive behaviour of malware[J]. *Journal of Information Security and Applications*, 2023, 74: 103451.
- [92] PENG F, DENG Z, ZHANG X, et al. X-Force: force-executing binary programs for security applications[C]// *Proceedings of the 23rd USENIX Security Symposium (USENIX Security)*. Berkeley: USENIX Association, 2014: 829-844.
- [93] YOU W, ZHANG Z, KWON Y, et al. PMP: cost-effective forced execution with probabilistic memory pre-planning[C]// *Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP 2020)*. Piscataway: IEEE Press, 2020: 1121-1138.
- [94] HE D N, XIE D C, WANG Y J, et al. Define-use guided path exploration for better forced execution[C]// *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*. New York: ACM, 2024: 287-299.
- [95] ZHAN D Y, TAN K, YE L, et al. An adversarial robust behavior sequence anomaly detection approach based on critical behavior unit learning[J]. *IEEE Transactions on Computers*, 2023, 72(11): 3286-3299.
- [96] GOYAL A, WANG G, BATES A. R-CAID: embedding root cause analysis within provenance-based intrusion detection[C]// *Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP 2024)*. Piscataway: IEEE Press, 2024: 3515-3532.
- [97] COSCIA A, LORUSSO R, MACI A, et al. APIARY: an API-based automatic rule generator for yara to enhance malware detection[J]. *Computers & Security*, 2025, 153: 104397.
- [98] UETZ R, HERZOG M, HACKLÄNDER L, et al. You can not escape me: detecting evasions of SIEM rules in enterprise networks[C]// *Proceedings of the 33rd USENIX Security Symposium (USENIX Security)*. Berkeley: USENIX Association, 2024: 5179-5196.
- [99] 杨秀璋, 彭国军, 刘思德, 等. 面向APT攻击的溯源和推理研究综述[J]. *软件学报*, 2025, 36(1): 203-252.
- YANG X Z, PENG G J, LIU S D, et al. Survey on attribution and inference research for APT attacks[J]. *Journal of Software*, 2025, 36(1): 203-252.

[作者简介]



王晨阳（1996-），男，河南新野人，武汉大学博士生，主要研究方向为恶意代码分析。



杨秀璋（1991-），男，贵州施秉人，贵州大学副教授，主要研究方向为威胁情报感知、恶意代码分析。



彭国军（1979-），男，湖北荆州人，武汉大学教授、博士生导师，主要研究方向为网络安全、信息系统安全。



周逸林（1998-），男，湖北荆州人，武汉大学博士生，主要研究方向为系统安全、固件安全。