

DOI:10.3969/j.issn.1671-0673. 2026. 04. 009

二进制代码语义恢复技术研究综述

孙佳佳, 李泽潭, 付艳华, 朱肖雅, 尹小康, 杨启超, 刘胜利

(信息工程大学, 河南 郑州 450001)

摘要: 二进制代码语义恢复技术旨在在缺失源代码与调试信息的条件下, 重建函数、变量、数据结构等高层语义信息, 以支撑恶意代码理解、漏洞定位与软件成分识别。受符号剥离、编译优化与跨平台差异影响, 高层语义与低层实现之间存在信息缺失与形态扰动并存的问题, 导致信息稀疏、表示异构及全局依赖难以刻画。系统回顾代表性研究, 将现有方法归纳为启发式规则匹配、深度学习判别式推断与大语言模型驱动的生成式恢复 3 类范式, 并梳理其演进脉络, 进一步总结主流数据集与评测指标的发展趋势。最后, 剖析了该领域研究面临的真实挑战, 并展望了未来研究方向。

关键词: 二进制代码; 语义恢复; 逆向工程; 代码理解; 深度学习; 大语言模型

中图分类号: TP393.1

文献标识码: A

文章编号: 1671-0673(2026)04-0443-07

A Survey on Binary Code Semantic Recovery Technology

SUN Jiajia, LI Zetan, FU Yanhua, ZHU Xiaoya, YIN Xiaokang, YANG Qichao, LIU Shengli

(Information Engineering University, Zhengzhou 450001, China)

Abstract: Binary code semantic recovery technology aims to reconstruct high-level semantic information such as functions, variables, and data structures in the absence of source code and debugging information, to support malicious code understanding, vulnerability localization, and software component identification. Due to symbol stripping, compiler optimizations, and cross-platform differences, there coexist problems of information loss and morphological perturbation between high-level semantics and low-level implementations, leading to sparse information, heterogeneous representations, and difficulties in characterizing global dependencies. Representative studies are systematically reviewed in this paper. The existing methods are categorized into 3 paradigms, which include heuristic rule matching, deep learning discriminative inference, and large language model-driven generative recovery, and their evolution is traced. The development trends of mainstream datasets and evaluation metrics are further summarized. Finally, the real challenges faced by research in this field are analyzed and future research directions are anticipated.

Key words: binary code; semantic recovery; reverse engineering; code comprehension; deep learning; large language model

二进制逆向工程是通过分析可执行文件来理解程序逻辑与运行机制的过程, 在安全审计、漏洞挖掘和遗留系统维护中具有重要意义^[1]。然而, 传统逆向分析高度依赖专家经验。

尽管 IDA Pro、Ghidra 等逆向分析工具在反汇编、反编译和控制流恢复方面已经取得了较好效果, 但其输出结果本质上仍属于对底层程序行为的近似重建, 难以完整保留源程序中的语义信

收稿日期: 2026-03-06; 修回日期: 2026-03-27

作者简介: 孙佳佳(2001—), 女, 硕士生, 主要研究方向为二进制代码语义恢复。E-mail: sunjiajia083@163.com

息^[2]。近期针对反编译能力的实证研究进一步表明,现有C反编译器在不同编译器和优化级别下仍存在明显的语义偏差与错误类型,这说明仅依赖反编译结果开展高层语义恢复仍面临输入噪声大、表示失真强和跨平台泛化不足等挑战。

为弥补上述不足,二进制语义恢复技术逐渐成为逆向工程领域的重要研究方向。相关研究已由早期基于启发式规则和程序分析的方法,发展到基于统计学习、深度学习以及生成式大语言模型的方法,研究对象也由函数名、变量名和类型信息恢复逐步扩展到功能摘要生成、数据结构推断和更高层次的程序行为理解。与此同时,国内研究者也围绕二进制安全分析、漏洞挖掘、代码表征和函数语义恢复等方向^[3-6]开展了系统研究,表明该领域正由单一程序分析问题演进为融合程序分析、表示学习与语义生成的综合研究方向。

基于此,本文对二进制语义恢复技术的发展脉络进行系统梳理,归纳不同技术路线的基本思想、代表性工作及其局限,并对未来发展趋势进行总结与展望。

1 语义恢复的分类

相较于源码分析,二进制分析在编译过程中丢失了大量高级结构与类型信息,导致底层指令与高层程序语义之间存在明显断层^[7]。二进制语义恢复的目标,就是从底层指令、控制流和数据流中重建这些丢失的高层语义信息。基于逆向工程的实际需求,本文将语义恢复归纳为以下4个核心任务。

1) 变量名与函数名恢复:用于揭示程序功能语义和开发者意图。

2) 基本类型恢复:推断变量、参数和返回值的基础类型,为数据流分析提供支持。

3) 结构体恢复:识别复合数据结构的大小、成员偏移及组织关系。

4) 控制流结构恢复:将底层跳转关系还原为高级语言中的分支、循环等结构。

2 基于启发式规则匹配的方法

早期二进制语义恢复主要依赖静态分析、动态分析和启发式规则匹配,通过控制流、数据流及指令依赖关系构建约束,以恢复类型信息、控制结构和数据布局。

2.1 基于静态约束的变量与类型推断

在类型恢复方面,2011年TIE^[8]率先将二进制类型恢复建模为静态约束求解问题,为后续研究奠定了基础,但其效果比较依赖指针分析精度。Retypd^[9]则在此基础上进一步发展出递归约束与子类型系统,将指针读写能力分离建模,从而缓解了复杂别名场景下的约束冲突问题。相比TIE,Retypd的核心创新不在于继续增加规则,而在于提升了类型系统本身对多态性、递归性与高优化机器码的表达能力。

总体上,在这一方向上,相关研究的演进并非简单增加规则,而是不断提升类型系统对复杂程序行为的表达能力。

2.2 基于图重构与动态监控的控制流及结构还原

在控制流恢复方面,DREAM^[10]通过语义保持的图重构方法,将非结构化控制流转化为更易理解的结构化形式,提高了反编译结果的可读性。

在数据结构恢复方面,REWARDS^[11]利用带时间戳的类型传播缓解运行时内存复用带来的歧义;随后,Howard^[12]在此基础上进一步面向数组与结构体长度恢复,引入基于栈帧与基指针相对访问模式的动态检测策略,将分析粒度从抽象内存对象推进到更具体的数据组织形式。DSI^[13]则继续向复杂动态数据结构场景扩展,不再局限于单个节点或单条访问轨迹,而是通过链与链图建模统一链接关系,并结合动态证据权重完成不规则结构分类。

总体上,这一方向的研究重点经历了从控制流可读性恢复扩展到复杂数据组织识别的演变。

2.3 基于概率推断与混合分析的复杂语义提取

随着分析对象日益复杂,部分研究开始引入概率推断与混合分析。OSPREY^[14]将类型和结构属性建模为随机变量,通过概率约束求解提升了结构推断覆盖率。OBJDIGGER^[15]则结合符号执行与跨过程数据流分析,实现了对C++对象成员和继承关系的恢复。

总体来看,基于启发式规则匹配的方法建立在静态约束、动态监控与概率推断等可解释分析机制之上,其优势在于推理链条明确、结果可验证性较强,尤其适用于类型、结构体布局与控制流等语义对象的精确恢复。然而,这类方法高度依赖专家设计的规则与程序分析器质量,当面对跨架构迁移、编译优化扰动以及大规模复杂程序时,往往存在规则扩展性不足与人工维护成本高的问

题。后续深度学习方法正是在这一背景下发展起来,其核心转变在于,不再显式枚举恢复规则,而是通过数据驱动方式学习底层表示与高层语义之间的映射关系,从而提升模型对复杂模式的自动抽取能力。为便于比较,表 1 总结了该阶段代表性方法的核心机制及适用场景。

表 1 基于启发式规则匹配的方法对比

预测任务	代表方法	年份	核心机制与特点	适用场景
内存布局与数据结构	REWARDS	2010	基于时间戳标签分配与双向类型传播消歧	结构布局重建、内存取证、模糊测试
基础变量类型	TIE	2011	基于子类型格的保守类型约束与全局求解	基础变量类型推断、静态控制流分析
数组与结构体	Howard	2011	追踪函数帧与指针步长,进行启发式数组检测	结构体/嵌套数组恢复、防御缓冲区溢出
C++对象与复杂结构	OBJDIGGER	2014	符号执行结合数据流分析,追踪 this 指针轨迹	C++对象恢复、自动提取构造函数与虚函数表
控制流结构	DREAM	2015	基于图切片的语义保持性转换	控制流梳理(消除 goto)、恶意软件逻辑重构
高级类型与多态	Retypd	2016	基于下推系统的递归约束,指针读写分离	多态/const 提取、处理高度优化的机器码
动态数据结构	DSI	2016	引入动态证据权重,提取满足链接条件的内存“链”	复杂/不规则内存分类(如链表、红黑树等)
变量类型	OSPREY	2021	内存属性随机变量建模,通过因子图与概率求解	复杂堆折叠恢复、对抗编译优化的变量恢复

3 基于深度学习的分类方法

随着二进制程序复杂度和数据规模不断增加,传统依赖人工规则的方法在可扩展性和维护成本上逐渐暴露局限。为此,研究者引入深度学习,将语义恢复建模为数据驱动的分类任务,通过学习底层表示与高层语义之间的映射关系提升恢复能力。

3.1 基于统计学习的函数语义推断

在函数语义恢复研究中,相关方法总体经历了从局部特征建模向全局语义建模的发展过程。早期工作主要基于指令模式、寄存器行为和内存访问等底层特征进行统计推断。DEBIN 结合极端随机树与条件随机场模型(Conditional Random Field, CRF)实现联合结构化预测。Jaffe 等^[16]借鉴统计机器翻译思想,将底层代码表示映射为高层语义标签。Punstrip^[17]进一步引入 Symbol2Vec,对函数调用关系中的上下文依赖进行向量化表示,推动函数语义分析从局部模式匹配转向上下文感知建模。

随后,EKLAVYA^[18]利用 Word2Vec 和门控循环单元(Gated Recurrent Unit, GRU)实现端到端的参数语义推断。Nero 通过构建增强调用站点图,利用图卷积网络(Graph Convolutional Network, GCN)直接对控制流拓扑与寄存器操作进行拓扑特征编码。SymLM^[19]则将这一思路扩展到跨过程层面,通过过程间控制流图(Interprocedural Control Flow Graph, ICFG)与上下文敏感嵌入显式建模调用者与被调用者之间的语义关联,使函数名预测从函数内部识别演进为过程间语义推断。

面向真实场景中的复杂问题,后续研究进一步从长尾分布、跨架构迁移和复杂编译扰动等方面开展优化。XFL^[20]面向真实数据集中的极端长尾分布问题,引入多标签学习框架,以提升未见标签下的泛化能力;Epitome^[21]则进一步面向跨架构与跨优化扰动,通过投票式分词与多任务学习增强语义稳定性。此外,MFNP^[22]利用 LLVM RetDec 将 x86、ARM 和 MIPS 等多架构二进制统一提升为中间表示,并结合基本块级控制流图与 Seq2Seq 模型实现跨架构函数名预测,缓解了底层指令差异带来的建模困难。

总体而言,该方向的技术路线已经由局部统计特征建模逐步发展为融合结构信息、过程间上下文、跨架构表示与生成式表达的综合建模范式。

3.2 基于序列联合建模的变量类型推断

在变量与类型恢复方面,TypeMiner^[23]代表了基于显式依赖图与传统分类器的早期路线,其主要价值在于证明了变量传播轨迹对于类型推断的重要性。CATI^[24]则针对“孤儿变量”这一特征稀疏问题,将关注点转向变量使用上下文,并引入卷积神经网络(Convolutional Neural Network, CNN)对局部语义片段进行建模从而补足传统依赖图难以覆盖的细粒度线索。随着恢复对象从基础标量扩展至复合数据类型,TYGR^[25]和 ByteTR^[26]分别通过门控图神经网络(Gated Graph Neural Network, GGNN)建模与跨函数图追踪,增强了模型对复杂类型关系与存储别名扰动的表达能力。

进一步,StateFormer^[27]率先尝试将执行过程建模为生成式状态序列,通过引入状态归一化机制,成功消除了编译优化导致的指令重排干扰,在细

粒度多级指针解析等极具挑战性的任务上表现卓越。然而,单一的时序建模仍难以实现命名与类型这两类强相关语义任务的互补。在此背景下,DIRTY^[28]的进一步创新则在于将变量命名与类型恢复进行联合优化,说明这两类语义任务之间并

非彼此独立,而是存在显著的互补关系。由此,该研究分支的演进逻辑可以概括为,从单变量、单任务、局部建模,逐步发展到跨函数、复合类型与联合语义恢复^[29]。上述代表性方法的核心架构与适用场合如表2所示。

表2 基于深度学习的分类方法对比

预测任务	代表方法	年份	核心架构与特征表示	解决的问题
函数参数与类型恢复	EKLAVYA	2017	指令嵌入+门控循环单元网络	端到端学习指令语义,推断函数参数数量与类型
符号与类型恢复	DEBIN	2018	特征工程+极端随机树+条件随机场	恢复符号与类型,处理局部预测冲突
函数名与变量名恢复	文献[16]	2018	反编译代码序列+统计机器翻译	恢复自然编程语义,弥补语义占位符缺失
基础类型与指针类型恢复	TypeMiner	2019	数据依赖图+多阶段分类	静态推断基础类型与指针类型
函数名恢复	Punstrip	2020	Symbol2Vec+条件随机场	引入调用上下文信息,增强函数语义表示
函数名恢复	Nero	2020	调用站点图+图卷积网络/长短期记忆网络/Transformer	建模函数结构信息,提升长函数命名能力
变量名与类型恢复	CATI	2020	变量使用上下文+卷积神经网络	缓解“孤儿变量”特征稀疏问题
多级指针与类型恢复	StateFormer	2021	状态序列表示+Transformer	抵抗指令重排干扰,解析多级指针语义
函数名恢复	SymLM	2022	过程间控制流图+上下文敏感嵌入	引入上下文语义,提升函数名预测效果
变量名与类型恢复	DIRTY	2022	预训练Transformer+联合学习	联合优化命名与类型恢复,增强语义一致性
函数名恢复	MFNP	2023	LLVM IR+基本块顺序指令语义+基本块级控制流图+序列模型	缓解多架构二进制表示差异,支持 x86/ARM/MIPS 跨架构函数名预测
函数名恢复	XFL	2023	多标签学习	缓解长尾分布问题,提升未见标签泛化能力
函数名恢复	Epitome	2024	投票分词+预训练图神经网络	增强跨架构、跨优化场景下的语义鲁棒性
复合数据类型恢复	TYGR	2024	异构图表示+图神经网络	精确建模复合类型,如结构体等
变量类型恢复	ByteTR	2025	跨函数变量图+门控图神经网络	缓解存储别名问题,提升跨函数类型恢复能力

4 基于大语言模型的生成式方法

由于早期深度学习方法多将语义恢复建模为封闭词表分类任务,因此在函数名预测等场景下容易受到词汇表外问题(Out-of-Vocabulary, OoV)的影响,即模型难以预测训练阶段未见过的新标签。相比传统深度学习方法,大语言模型具备开放词表生成能力,能够突破固定标签空间限制,直接生成更符合程序语义的高层符号。因此,二进制语义恢复研究正逐步从分类范式转向生成范式。

4.1 基于跨模态对齐的指令表征

大语言模型应用于二进制语义恢复的关键前提,是将底层二进制表示映射到语言模型可理解的语义空间。跨模态对齐的目标,正是在二进制表示与自然语言表示之间建立稳定关联。

在这一方向上,BinAdapter^[30]通过持续学习机制增强模型对新架构指令的适应能力。llasm^[31]采用编码器—解码器架构,先利用专用编码器提取汇编语义特征,再通过投影层映射到语言模型输

入空间,从而缓解代码模态与文本模态之间的差异。总体来看,跨模态对齐是大语言模型理解二进制语义的关键基础,其关键不在于简单转换输入形式,而在于缓解底层指令表示与自然语言语义空间之间的模态鸿沟,使语言模型能够在保持执行语义的同时生成可解释的高层符号。

4.2 基于偏好对齐的语义符号生成

在具备语义理解能力后,生成结果是否符合开发者命名习惯与逆向分析需求,成为另一关键问题^[32]。偏好对齐的目标,使模型输出不仅语义相关,而且风格自然、粒度合理。与跨模态对齐侧重能否理解不同,偏好对齐更关注生成结果是否符合真实开发语境与逆向分析需求。

GENNM^[33]通过引入双向调用上下文和偏好优化机制,提升了命名结果的语义合理性与风格一致性。BLens^[34]则通过对比学习拉近汇编表示与文本描述之间的语义距离,增强了函数名称生成能力。该方向表明,生成式语义恢复正在向生成更符合人类习惯的命名方式发展。

4.3 基于混合推理的数据结构恢复

尽管大语言模型在语义归纳和开放生成方面具有优势,但在大型结构体恢复和跨函数关系分析中,仅依赖神经网络难以保证全局一致性。尤其在大型结构体恢复、跨字段依赖分析和不完整数据流场景中,局部生成结果难以直接满足程序结构约束。因此,需要借助逻辑规则或约束求解实现全局一致性校验。部分研究将语言模型与符号逻辑、约束求解等方法结合,形成混合推理框架。

ReSym^[35]将 LLM 生成的变量和字段候选结果

交由 Prolog 逻辑引擎进一步消解与校验,从而结合局部语义预测与全局结构约束,提升复杂数据结构恢复的准确性。HyRES^[36]进一步在 LLVM 中间表示 (LLVM Intermediate Representation, LLVM IR) 上结合 Datalog 演绎,增强了跨架构和数据流不完整场景下的结构一致性。这类方法的核心在于将神经生成与符号约束相结合,形成候选生成、规则验证与全局一致性修正的统一框架,从而提升恢复结果的逻辑一致性与可验证性。本节涉及的基于大语言模型的生成式语义恢复方法对比如表 3 所示。

表 3 基于大语言模型的生成式方法对比

预测任务	代表方法	年份	模型	解决的问题
函数名恢复	BinAdapter	2024	持续学习+Transformer	缓解新架构指令适配中的灾难性遗忘问题
函数名恢复	llasm	2025	jTrans 编码器+线性投影对齐+Llama2	缓解代码模态与文本模态不对齐问题,提升未见项目泛化能力
变量名恢复	GENNM	2025	CodeGemma+Llama+双向调用上下文+偏好优化	缓解开放词表场景下的变量命名问题,提升语义合理性
变量与字段恢复	ReSym	2024	StarCoder+Prolog 逻辑约束	重建自定义数据结构的内存布局,提升全局一致性
函数名恢复	BLens	2025	CoCaGIT+对比学习	缓解跨项目分布偏移,增强函数命名能力
复杂数据结构恢复	HyRES	2026	LLVM IR+Datalog 演绎+Qwen2-72B	识别间接调用场景下的复杂结构体关系,增强结构一致性
多任务逆向分析辅助	ReCopilot	2025	Qwen2.5-Coder+任务协同生成	减少对专家的依赖,抑制模型幻觉
宏观程序结构恢复	BinStruct	2025	LLM+高层结构推断	恢复文件级或模块级程序组织语义

5 面临挑战与未来展望

尽管生成式方法推动了二进制语义恢复的发展,但在工业软件、恶意代码和物联网固件等复杂场景下,相关研究仍面临诸多挑战。二进制分析的困难不仅来自模型能力有限,更与控制流不确定性、数据流复杂性以及编译优化扰动等底层因素密切相关。

5.1 当前技术面临的核心挑战

1)间接跳转导致控制流恢复困难。间接跳转目标依赖运行时状态,难以仅通过静态分析准确解析,进而影响函数边界识别和后续语义推断。

2)内存别名带来数据流分析复杂性。内存别名问题使静态分析常需采用保守近似,容易扩大分析范围,降低恢复精度与可扩展性。

3)编译优化模糊程序原始语义边界。函数内联、寄存器重用和代码重排等优化模糊了原始语义边界,增加了函数名、变量和类型恢复的难度^[37]。

5.2 未来研究方向

1)面向混淆与对抗变异的鲁棒语义恢复。真

实场景中的恶意代码和固件程序常伴随加壳、控制流平坦化和垃圾指令插入等混淆手段,未来可进一步研究鲁棒表征方法,以及去混淆技术与语义恢复模型的协同机制。

2)动静结合的混合分析机制。针对间接跳转和内存别名等静态分析难以准确处理的问题,未来可结合动态执行、沙箱监测和模糊测试等手段,引入运行时信息与静态表征融合,以提升语义恢复的准确性。

6 结束语

二进制代码语义恢复旨在重建函数名、变量名、结构体等高层语义信息。该领域的核心困难不仅在于语义生成本身,更在于间接跳转、内存别名、编译优化和代码混淆等底层因素带来的分析障碍。未来,推动静态分析、动态分析与智能模型的协同融合,将是提升二进制语义恢复实用性与鲁棒性的关键方向。

参考文献:

- [1] Shoshitaishvili Y, Wang R Y, Salls C, et al. Sok: (state of) the art of war: offensive techniques in binary analysis [C]//Proceedings of the 2016 IEEE symposium on security and privacy. Piscataway, USA: IEEE, 2016:138-157.
- [2] David Y, Alon U, Yahav E. Neural reverse engineering of stripped binaries using augmented control flow graphs [C]//Proceedings of the ACM on programming languages. New York, USA: ACM, 2020,4(OOPSLA):225.
- [3] 周忠君,董荣朝,蒋金虎,等.二进制代码安全分析综述[J].计算机系统应用,2023,32(1):1-11.
- [4] 王夏菁,胡昌振,马锐,等.二进制程序漏洞挖掘关键技术研究综述[J].信息安全,2017(8):1-13.
- [5] 田硕,梁洪亮.二进制程序安全缺陷静态分析方法的研究综述[J].计算机科学,2009,36(7):8-14.
- [6] 郑炜,唐辉,陈翔,等.基于深度学习的函数名一致性检查及推荐方法[J].软件学报,2023,35(10):4604-4622.
- [7] He J X, Ivanov P, Tsankov P, et al. Debin: predicting debug information in stripped binaries [C]//Proceedings of the 2018 ACM SIGSAC conference on computer and communications security. New York, USA: ACM, 2018:1667-1680.
- [8] Lee J H, Avgerinos T, Brumley D. TIE: Principled reverse engineering of types in binary programs [C]//Network and distributed system security symposium, 2011. DOI: 10.1184/R1/6469466.V1.
- [9] Noonan M, Loginov A, Cok D. Polymorphic type inference for machine code [C]//Proceedings of the 37th ACM SIGPLAN conference on programming language design and implementation. New York, USA: ACM, 2016:27-41.
- [10] Yakdan K, Eschweiler S, Gerhards-Padilla E, et al. No more gotos: decompilation using pattern-independent control-flow structuring and semantic-preserving transformations [C]//Network and distributed system security symposium, 2015. DOI: 10.14722/NDSS.2015.23185.
- [11] Lin Z Q, Zhang X Y, Xu D Y. Automatic reverse engineering of data structures from binary execution [C]//Proceedings of the 11th annual information security symposium (CERIAS'10). West Lafayette, USA: CERIAS-Purdue University, 2010:5.
- [12] Slowinska A, Stancescu T, Bos H. Howard: a dynamic excavator for reverse engineering data structures [C/OL]//NDSS 2011. <https://www.ndss-symposium.org/ndss2011/howard-a-dynamic-excavator-for-reverse-engineering-data-structures>.
- [13] White D H, Rupprecht T, Lüttgen G. DSI: an evidence-based approach to identify dynamic data structures in C programs [C]//Proceedings of the 25th international symposium on software testing and analysis. New York, USA: ACM, 2016:259-269.
- [14] Zhang Z, Ye Y P, You W, et al. Osprey: recovery of variable and data structure via probabilistic analysis for stripped binary [C]//Proceedings of the 2021 IEEE symposium on security and privacy. Piscataway, USA: IEEE, 2021:813-832.
- [15] Jin W, Cohen C, Gennari J, et al. Recovering C++ objects from binaries using inter-procedural data-flow analysis [C]//Proceedings of ACM SIGPLAN on program protection and reverse engineering workshop 2014. New York, USA: ACM, 2014:1-11.
- [16] Jaffe A, Lacomis J, Schwartz E J, et al. Meaningful variable names for decompiled code: a machine translation approach [C]//Proceedings of the 26th conference on program comprehension. Piscataway, USA: IEEE, 2018:20-30.
- [17] Patrick-Evans J, Cavallaro L, Kinder J. Probabilistic naming of functions in stripped binaries [C]//Proceedings of the 36th annual computer security applications conference. New York, USA: ACM, 2020:373-385.
- [18] Chua Z L, Shen S Q, Saxena P, et al. Neural nets can learn function type signatures from binaries [C]//Proceedings of the 26th USENIX conference on security symposium. New York, USA: ACM, 2017:99-116.
- [19] Jin X, Pei K X, Won J Y, et al. SymLM: predicting function names in stripped binaries via context-sensitive execution-aware code embeddings [C]//Proceedings of the 2022 ACM SIGSAC conference on computer and communications security. New York, USA: ACM, 2022:1631-1645.
- [20] Patrick-Evans J, Dannehl M, Kinder J. XFL: naming functions in binaries with extreme multi-label learning [C]//Proceedings of the 2023 IEEE symposium on security and privacy. Piscataway, USA: IEEE, 2023:2375-2390.
- [21] Zhang X L, Xu Z Z, Yang S G, et al. Enhancing function name prediction using votes-based name tokenization and multi-task learning [C]//Proceedings of the ACM on software engineering. New York, USA: ACM, 2024:1679-1702.
- [22] 邵文强,蔡瑞杰,宋恩舟,等.基于语义的多架构二进制函数名预测方法[J].计算机科学,2023,50(10):369-376.
- [23] Maier A, Gascon H, Wressnegger C, et al. TypeMiner: recovering types in binary programs using machine learning [C]//Detection of intrusions and malware, and vulnerability assessment. Cham, Switzerland: Springer, 2019:288-308.
- [24] Chen L G, He Z L, Mao B. CATI: context-assisted type inference from stripped binaries [C]//Proceedings of the 2020 50th annual IEEE/IFIP international conference on dependable systems and networks. Piscataway, USA: IEEE, 2020:88-98.
- [25] Zhu C, Li Z Y, Xue A, et al. TYGR: type inference on

- stripped binaries using graph neural networks [C]//Proceedings of the 33rd USENIX security symposium. Berkeley, USA: USENIX Association, 2024:4283-4300.
- [26] Li G Y, Shang X W, Cheng S Y, et al. Beyond the edge of function: unraveling the patterns of type recovery in binary code [PP/OL]. arXiv. (2025-05-04) [2026-07-23]. <https://doi.org/10.48550/arXiv.2503.07243>.
- [27] Pei K X, Guan J, Broughton M, et al. StateFormer: fine-grained type recovery from binaries using generative state modeling[C]//Proceedings of the 29th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering. New York, USA: ACM, 2021:690-702.
- [28] Chen Q, Lacomis J, Schwartz E J, et al. Augmenting decompiler output with learned variable names and types [C]//Proceedings of the 31st USENIX security symposium. Berkeley, USA: USENIX Association, 2022:4327-4343.
- [29] Caballero J, Lin Z Q. Type inference on executables[J]. ACM Computing Surveys (CSUR), 2016,48(4):1-35.
- [30] Murodova N, Koo H. BinAdapter: leveraging continual learning for inferring function symbol names in a binary [C]//Proceedings of the 19th ACM Asia conference on computer and communications security. New York, USA: ACM, 2024:1200-1213.
- [31] Sha Z H, Wang H, Gao Z Y, et al. Llasm: naming functions in binaries by fusing encoder-only and decoder-only LLMs[J]. ACM Transactions on Software Engineering and Methodology, 2025,34(4):1-22.
- [32] Dong Y H, Ding J Z, Jiang X, et al. CodeScore: evaluating code generation by learning code execution[J]. ACM Transactions on Software Engineering and Methodology, 2025,34(3):1-22.
- [33] Xu X Z, Zhang Z, Su Z A, et al. Unleashing the power of generative model in recovering variable names from stripped binary[C]//Proceedings of the 2025 network and distributed system security symposium. Rosten, USA: Internet Society, 2025. DOI: 10.14722/ndss.2025.240276.
- [34] Benoit T, Wang Y R, Dannehl M, et al. BLens: contrastive captioning of binary functions using ensemble embedding[C]//Proceedings of the 34th USENIX security symposium. Berkeley, USA: USENIX Association, 2025:6877-6896.
- [35] Xie D N, Zhang Z, Jiang N, et al. ReSym: harnessing LLMs to recover variable and data structure symbols from stripped binaries[C]//Proceedings of the 2024 on ACM SIGSAC conference on computer and communications security. New York, USA: ACM, 2024:4554-4568.
- [36] Sha Z H, Shu H, Wang H, et al. HyRES: recovering data structures in binaries via semantic enhanced hybrid reasoning[J]. ACM Transactions on Software Engineering and Methodology, 2026,35(3):1-28.
- [37] Sirlanci M, Yagemann C, Lin Z Q. An empirical study of C decompilers: performance metrics and error taxonomy [C]//Proceedings of the 20th ACM asia conference on computer and communications security. New York, USA: ACM, 2025:1707-1723.

(编辑: 邵 华)

~~~~~

(上接第435页)

- [16] Bellini E, Gerault D, Hambitzer A, et al. A cipher-agnostic neural training pipeline with automated finding of good input differences [J]. IACR Transactions on Symmetric Cryptology, 2023:184-212.
- [17] Wang G, Wang G L. Enhanced related-key differential neural distinguishers for SIMON and SIMECK block ciphers[J]. PeerJ Computer Science, 2024,10:e2566.
- [18] Hou Y F, Liu J, Han S X, et al. Improving deep learning-based neural distinguisher with multiple ciphertext pairs for speck and Simon [J]. Scientific Reports, 2025, 15: 13696.
- [19] Zhao Z H, Jia Y X, Zhang Z H. Multi-scale convolution multi-graph attention neural networks for traffic flow forecasting [C]//Proceedings of the 2024 16th international conference on machine learning and computing. New York, USA: ACM, 2024:176-184.
- [20] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need [C]//Advances in neural information processing systems. Red Hook, USA: Curran Associates Inc., 2017: 5998-6008.
- [21] He K M, Zhang X Y, Ren S Q, et al. Deep residual learning for image recognition [C]//Proceedings of the 2016 IEEE conference on computer vision and pattern recognition. Piscataway, USA: IEEE, 2016:770-778.
- [22] Chen Y, Shen Y T, Yu H B. Neural-aided statistical attack for cryptanalysis [J]. The Computer Journal, 2023, 66(10):2480-2498.
- [23] Hong D, Lee J K, Kim D C, et al. LEA: a 128-bit block cipher for fast encryption on common processors[C]//Proceedings of the Information security applications. Cham, Switzerland: Springer, 2014:3-27.
- [24] Needham R M, Wheeler D J. Tea extensions [R]. Cambridge, England: Computer Laboratory, University of Cambridge, 1997.

(编辑: 冯 春)