基于自动化动态库劫持结合 Datalog 的堆漏洞检测系统设计

罗治祥[1] 赖怡聪[1] 曹佳伟[3] 李乐言[1,2*]
LUO Zhixiang LAI Yicong CAO Jiawei LI Leyan

摘 要 堆漏洞作为二进制程序中的重要安全威胁，长期以来受到安全研究者广泛关注。传统的静态分析与动态测试在检测精度、扩展性与适配性方面仍存在诸多不足。基于此，文章提出一种结合自动化动态库劫持与 Datalog 规则匹配的数据流分析方法，实现对堆漏洞（包括 Use-After-Free、Double Free、堆溢出等）的高效、精准检测。通过运行时劫持内存操作函数（如 malloc、free），实时收集堆内存变化信息，并使用可扩展的 Datalog 规则进行漏洞检测分析。实验结果表明，所提方法具有良好的检测精度与可移植性，能有效适应多平台环境并支持新型漏洞规则的扩展。

关键词 堆漏洞；动态库劫持；Datalog；Use-After-Free；动态分析；自动化漏洞检测

doi：10.3969/j.issn.1672-9528.2026.06.034

0 引言

近年来，随着软件系统复杂度的不断提高以及软件部署场景的日益多样化，软件漏洞的数量与影响范围呈现出爆发式增长趋势。尤其是在操作系统核心组件、浏览器引擎、车载系统、智能硬件固件等领域，由于开发周期紧迫、安全测试不完善以及安全意识缺失等问题，导致诸如堆栈溢出、Use-After-Free（UAF）、Double-Free 等内存相关漏洞[1]频发。堆内存漏洞作为攻击者实现远程代码执行、提权、沙箱逃逸等攻击的重要入口，其检测与修复已成为软件安全领域的重要课题。

为此，研究人员提出了诸多软件漏洞检测技术，整体

1. 工业和信息化部电子第五研究所 广东广州 511370
2. 智能产品质量评价与可靠性保障技术工业和信息化部重点实验室 广东广州 511370
3. 一汽大众汽车有限公司 吉林长春 130011

未来可以从两方面进一步优化：一是引入非对称密钥动态更新机制，结合业务场景定期更新 SM2 公钥与私钥，提升长期运行的安全性；二是引入类 CA 认证机制完成客户端身份校验。

参考文献：

[1] 国家密码管理局 .SM2 密码算法加密签名消息语法规范 :GM/T 0010—2023[S] 北京 : 国家密码管理局 ,2023.
[2] 李露 , 谢映宏 , 李蔚凡 , 等 . 基于国密算法的配电网终端通信安全架构研究 [J]. 浙江电力 ,2022,41(12):79-87.
[3] 康剑萍 , 王沈敏 , 杜竹青 .PKI 技术在信息安全中的应用 [J]. 自动化仪表 ,2020,41(4):107-110.
[4] 国家市场监督管理总局 , 中国国家标准化管理委员会 . 电力监控系统网络安全防护导则 :GB/T 36572-2018 [S]. 北京 : 中国国家标准化管理委员会 ,2018.
[5] 国家密码管理局 .SM4 分组密码算法 :GM/T 0002-2012[S]. 北京 : 国家密码管理局 ,2012.
[6] 邓大为 , 李可 , 陆俊 . 基于 CIM/E 文件的电网全景建模技术研究 [J]. 广东电力 ,2013,26(11):49-53.
[7] 申昕怡 , 宋广彦 . 电力调度数据的网络传输技术与安全渗透分析 [J]. 电气技术与经济 ,2024(2):112-114.
[8]Khlebnikov A, Adolfsen J. Demystifying cryptography with OpenSSL 3.0: discover the best techniques to enhance your network security with OpenSSL 3.0[M]. Birmingham—Mumbai: Packt Publishing, 2022.
[9]Dey N. Cross-platform development with Qt 6 and modern C++: design and build applications with modern graphical user interfaces without worrying about platform dependency[M]. Birmingham: Packt Publishing, 2021.
[10]Lazarus G. Mastering Qt 5:create stunning cross-platform applications using C++ with Qt widgets and QML with Qt quick[M]. Birmingham: Packt Publishing, 2016.

【作者简介】

陈风帆（1995—），男，湖南怀化人，本科，高级工程师，研究方向：信息化技术在电网中的应用。

（收稿日期：2025-11-25 修回日期：2026-06-02）

可分为两大类：静态分析 [2] 与动态测试 [3]。静态分析具有覆盖率高、无需执行等优点，但在处理二进制程序、混淆代码或压缩代码时，常面临难以还原真实控制流、数据流的困难，尤其是在缺乏源代码的条件下，分析精度与效率大打折扣；动态测试方法，基于模糊测试 [4]（Fuzzing）与符号执行（Symbolic Execution）的方案，在处理未知漏洞检测方面表现出良好的自动化能力，但也受到路径爆炸、低代码覆盖率以及高误报等问题的困扰。

针对以上问题，现有的一些工作尝试融合静态与动态技术，构建更为精准的漏洞检测模型。其中，一种较为新颖的思路是引入逻辑程序语言 Datalog[5]，以其声明式逻辑推理能力，对程序运行过程中的状态信息进行关系建模，从而实现漏洞模式匹配。例如，谭添等 [6] 提出通过将程序执行事件映射为逻辑事实集合（Facts），并在其上定义指针分析规则，转入到 Datalog 推理引擎中进行分析，获取指针分析以及污点分析结果，为堆内存漏洞分析打下坚实基础。

尽管目前 Datalog 在软件分析中的理论优势已逐渐显现，但其在真实系统中的部署仍面临诸多挑战。首先，如何实时获取二进制程序运行时的内存行为数据并转化为高质量 Datalog 输入事实，是基于 Datalog 构建堆内存漏洞分析系统可用性的关键；其次，如何保证日志采集对目标程序的侵入性小、性能影响可控，亦是系统工程设计的重要目标。

为有效地解决上述问题，本文设计并实现了一种以动态库劫持技术 [7] 为基础，以 Datalog 逻辑推理为核心的堆漏洞自动化检测框架。其基本思路是：构建可跨平台部署的动态链接库，在目标程序执行期间对 malloc、free、realloc 等关键的堆操作函数实施拦截，同步采集操作参数、调用上下文及执行路径等相关信息。随后再将上述日志数据规范化地映射为 Datalog 逻辑事实，结合可自定义配置的规则集，借助高性能推理引擎完成对 UAF、DoubleFree、堆溢出等典型漏洞的自动识别。

与 angr[8]、BinAbsInspector[9] 等主流静态分析工具相比，上述工具普遍面临检测规则难以扩充、环境配置繁琐等工程层面的制约。本方案在部署复杂度、策略灵活性和执行效率上均具备明显优势，有效提升了系统的工程实用价值。本研究在理论上验证了 Datalog 机制在动态漏洞检测场景中的可行性，同时也为高可扩展、跨平台的二进制漏洞分析系统建设提供了可参考的技术路径。

1 方法设计

1.1 系统整体架构

本文所提出的检测系统将动态库劫持技术与 Datalog 推理引擎进行了有机结合，形成了一整套轻量、高效且便于扩展的堆漏洞分析体系。该系统在架构层面划分成了四个功能模块，分别承担日志采集、数据预处理、规则推理与结果输出等职责，各模块之间的协作关系如图 1 所示。

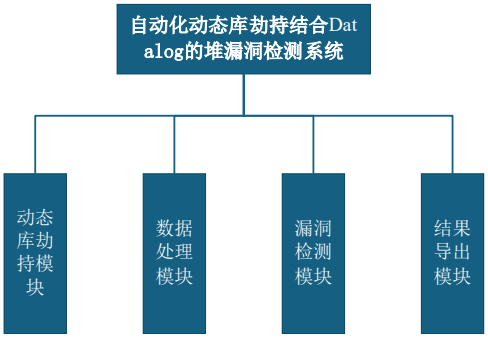

图 1 系统结构图

各模块的具体功能如下所示：

动态库劫持模块：通过自定义拦截函数并巧妙借助自动化脚本，生成适配多平台的动态库文件（.so），从而实现对目标程序运行过程中的 malloc、calloc、free、realloc 等堆操作函数的透明拦截。

数据处理模块：对采集到的原始内存操作记录进行清洗与结构化转换，滤除冗余字段，最终生成符合 Datalog 语法规范的逻辑事实（facts），以供后续推理使用。

漏洞检测模块：以 pyDatalog 等推理引擎作为执行载体，针对预定义的漏洞检测规则对输入事实进行模式匹配，最终输出各类漏洞的详细分析结果。

结果导出模块：汇总系统的推理结论，生成可视化图表或结构化的文本报告，还支持 PDF 导出与命令行展示等多种多样的输出形式。

该系统架构具备良好的平台兼容性，可以适配 x86、ARM 等多种主流指令集架构，同时还支持用户按需扩充检测规则，从而具备了快速响应新型漏洞检测需求的能力。

1.2 动态库劫持机制设计

动态库劫持是本系统设计方法的核心组成之一。通过在程序运行时替换其默认使用的动态链接库函数，使得系统可无侵入式地捕获堆内存分配与释放行为，形成一套轻量化、低开销的数据采集方案。

1.2.1 函数劫持目标

主要劫持的函数包括：

malloc(size_t size)、calloc(size_t n, size_t size)、realloc(void *ptr, size_t size) 等堆内存分配系列函数；

free(void *ptr) 等堆内存释放系列函数；

read(int fd, void *buf, size_t count)、write(int fd, const void *buf, size_t count) 等数据访问系列函数；

此外，还可扩展劫持如 memcpy、memmove 等内存拷贝

函数，以进一步捕捉堆上缓冲区操作行为，为后续堆溢出与重叠写入分析提供支持。

1.2.2 劫持机制实现

在 Linux 平台上，采用 LD_PRELOAD[10] 机制进行劫持。此机制允许用户在程序运行时预加载一个共享库，该共享库中定义的同名函数将替代系统库中原始函数，从而实现对特定函数的劫持。

自定义函数内部逻辑包含以下三步：

（1）记录操作参数与调用时序信息，如分配大小、返回地址、调用栈等；

（2）调用真实函数（通过 dlsym(RTLD_NEXT, "malloc") 等方法获取）；

（3）输出日志记录到共享内存或数据库，格式化为统一结构，便于后续分析。

示例代码如下：

```
void* malloc(size_t size) {
    void* (*real_malloc)(size_t) = dlsym(RTLD_NEXT, “malloc”);
    void* ptr = real_malloc(size);
    log_to_db(“malloc”, ptr, size, get_timestamp(), get_stack_trace());
    return ptr;
}
```

1.2.3 劫持信息格式与结构

为了统一记录格式并便于后续解析，定义如下日志结构和对应示例记录：

OPERATION_TYPE | TIMESTAMP | CHUNK_ADDR | SIZE | STACK_TRACE

malloc | 1628563920 | 0x55555575b040 | 100 | main->init->allocate

free | 1628563925 | 0x55555575b040 | NULL | main->cleanup

日志记录支持输出至本地文件、SQLite 数据库或 Redis 等内存型数据库，并支持添加额外字段，如线程 ID、调用模块、执行路径等，以支持后续的多维分析。

1.3 数据处理与 Datalog 事实生成

从劫持模块收集的原始日志文件并不适合直接用于 Datalog 推理，因此需经过结构化转换过程。此过程包括日志清洗、事实抽取与规则映射三步。

1.3.1 日志清洗与过滤

原始日志中可能存在重复记录（如内联函数调用多次记录）、无效指令（如栈地址访问）或测试产生的伪操作等。需先进行如下预处理：

（1）删除重复记录（使用 chunk_addr+timestamp 哈希去重）；

（2）补全未采集的操作参数（如从前后日志推测 write 操作大小）。

此外，系统还设计了正则匹配模板，自动提取操作参数信息，并对日志结构中存在的格式偏差进行自动修正，提升日志质量。

1.3.2 结构化为 Datalog 事实

Datalog 以关系型数据库逻辑为基础，使用逻辑谓词（predicate）表示实体关系。将每一条操作转换为如下形式：

op(OperationID,OpType,Address,Size)，其中 OperationID 为操作步骤位置，可以对应为代码地址。若处于循环中，亦可添加参数进行标识；OpType 为操作类型，如 read、write、malloc 以及 free 等函数调用类型；Address 为当前操作的堆块地址；Size 为当前操作的堆块大小。例如：Op(1,“malloc”, 0x55555575b040, 100)。

同时，可增加上下文信息，如栈地址、线程 ID、模块名等，以支持更复杂的规则建模。

1.3.3 关系建模与规则映射

构建以下核心关系用于建模堆内存行为状态转移：

state(Address, OperationID, Status)：指示某堆地址在某一操作后的状态（如激活、释放等）。

double_free(Address, OperationID)：两次释放指令判定条件。

uaf(Address, OpType, OperationID)：在已释放内存上的读写操作。

heap_overflow(Address,OpType,OperationID,ChunkSize,OpSize)：检测越界写入操作是否超出原 chunk 大小。

规则模板如下：

```
state(Addr, Op, ‘activate’) <= (
  op(Op, ‘malloc’, Addr, Size)
)
state(Addr, Op, ‘freed’) <= (
  op(Op, ‘free’, Addr, None)
)
state(Addr, Op, Status) <= (
  op(Op, OpType, Addr, Size)
  & (OpType._in(["read", "write"]))
  & state(Addr, Op-1, Status)
)
double_free(Addr, Op) <= (
  op(Op, ‘free’, Addr, None)
    &state(Addr,Op-1,’freed’)
)
```

```
UAF(Addr,OpType,Op) <= (
  op(Op, OpType, Addr, Size)
    &(OpType._in([“read”, “write”]))
    &state(Addr,Op-1,’freed’)
)
 heap_overflow(Addr, OpType, Op,ChunkSize,OpSize) <= (
  op(Op, OpType, Addr, OpSize)
    & (OpType._in([“read”, “write”]))
    &state(Addr, PreOp, ‘activate’)
    & (PreOp < Op)
     &op(MaxPreOp,’malloc’,Addr,ChunkSize) &
 (MaxPreps == max(item[0] for item in op(PreOp, ‘malloc’,
 Addr, PreSize)))
&(ChunkSize < OpSize)
)
```

1.4 扩展能力设计

为适应未来的新型漏洞类型检测需求以及更加深层次的语义分析场景，本系统在扩展性方面进行了以下设计：

（1）规则热更新与多规则并行支持：允许用户在运行时动态挂载多个规则文件，系统自动完成加载与规则融合，无需再进行重启或者重新编译。

（2）上下文语义增强：通过解析 DWARF 等调试信息引入源码符号表，将内存操作步骤与函数名、源文件以及行号等信息建立关联，提升分析结果的可读性与定位精准度。

综合以上设计，本系统实现了从无侵入式日志采集、事实建模、逻辑推理到报告输出的全流程自动化闭环，为堆漏洞检测研究方向提供了一种可移植性、可扩展性并且兼顾效率的工程化解决方案。

2 结论与展望

本文提出了一种融合自动化动态库劫持技术与 Datalog 规则推理的堆漏洞检测系统设计方案，在保障检测的精度与效率前提下，同时也兼顾了良好的扩展能力与跨平台适用性。后续的研究将进一步探索污点分析、控制流图信息及时序上下文的融合引入，用以增强系统对复杂漏洞场景的分析能力，并且将开展针对物联网设备与嵌入式系统环境的进一步部署适配研究。

参考文献：

[1]Aleph One. Smashing the stack for fun and profit[J]. Phrack Magazine, 1996, 7(49): 14-16.

[2]Chess B, McGraw G. Static analysis for security[J]. IEEE Security & Privacy, 2004, 2(6): 76-79.

[3]Gosain A, Sharma G. A survey of dynamic program analysis techniques and tools[C]//Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA)2014 .Berlin:Springer,2014: 113-122.

[4]Sutton M, Greene A, Amini P. Fuzzing: brute force vulnerability discovery[M]. Upper Saddle River: Addison-Wesley, 2007.

[5]Ceri S, Gottlob G, Tanca L. What you always wanted to know about Datalog (and never dared to ask)[J]. IEEE Transactions on Knowledge and Data Engineering, 1989, 1(1): 146-166.

[6] 谭添 , 马晓星 , 许畅 , 等 .Java 指针分析综述 [J]. 计算机研究与发展 , 2023,60(2):274-293.

[7]Lu Hongjiu. ELF: from the programmer's perspective[R]. 1995-05-17.http://www.skyfree.org/linux/references/ELF_Programmer.pdf.

[8]Shoshitaishvili Y, Wang Ruoyu, Salls C, et al. SoK: (state of) the art of war: offensive techniques in binary analysis[C]//2016 IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE, 2016: 138-157.

[9]Keen Security Lab,Tencent. BinAbsInspector: vulnerability scanner for binaries [EB/OL].(2022-04-20)[2026-05-12]. https://github.com/KeenSecurityLab/BinAbsInspector.

[10]Pulo K. Fun with LD_PRELOAD[C/OL]//Proceedings of the 2009 Linux Conference Australia. Canberra: Australian National University, 2009[2026-05-17].https://www.kev.pulo.com.au/publications/lca2009/lca2009-kevin-pulo-fun-with-ld_preload.pdf.

【作者简介】

罗治祥（1998—），男，四川遂宁人，本科，研究方向：漏洞挖掘、网络安全、车联网信息安全，email:luozhixiang@ceprei.biz。

赖怡聪（1995—），男，广东河源人，本科，助理工程师，研究方向：汽车信息安全、物联网漏洞安全、车联网体系安全，emali: laiyicong@ceprei.biz。

曹佳伟（1993—），男，四川眉山人，本科，研究方向：企业安全运营体系、网络安全攻防、汽车信息安全，emali: cjwzt@foxmail.com。

李乐言（1993—），通信作者（email:liyueyan@ceprei.biz），男，四川广安人，本科，工程师，研究方向：信息安全攻防技术和质量评价技术、智能产品及汽车信息安全。

（收稿日期：2025-10-15 修回日期：2026-05-28）