



腾讯安全沙龙

第8期 铸刃止戈 以智御危

分享人：彭佳仁

从架构分析到实测： LLM自动渗透测试实证研究

Acknowledgement

四川大学 DAS-Lab, 清华大学 NISL-VUL337, 腾讯云安全;

以及其他高校各位作者 (Renyang Liu, Haoran Ou, Yuqiang Sun, Jiancheng Zhang, Fan Shi, Hongda Sun, Rui Yan) 的付出。

LLM自动化渗透测试的实证研究

目录

海
纳
百
川
·
有
容
乃
大



● PART 1 / 研究背景

● PART 2 / 系统性梳理

● PART 3 / 实证研究

● PART 4 / 总结与展望



1 背景与意义

背景与意义

现有工作的空白

- 缺乏对基于大语言模型（LLM）的自动化渗透测试框架（AutoPT）的系统性架构分析
- 缺乏在统一基准下的大规模实证比较
- 以往的工作集中在深度强化学习的方法，而非基于LLM的范式
- 仅停留在宏观层面的分析，没有细粒度的架构解构

我们的贡献

- 首个关于基于LLM的AutoPT的系统化知识，**6维架构分类**
- 采用统一基准对**13个开源框架和2个基线框架**进行了实证评估
- 提出了超过**10个**关键实证发现

现有AutoPT框架是如何实现的？
现有AutoPT存在什么问题？
在制作AutoPT框

2 系统性梳理

01

智能体架构

- 角色定义
- 单 vs. 多智能体

02

智能体规划

- 基于线性、基于树、基于图的规划
- 反馈策略

03

智能体记忆

- 记忆压缩
- 记忆结构

04

智能体执行

- 执行角色
- 工具选择
- 工具调用

05

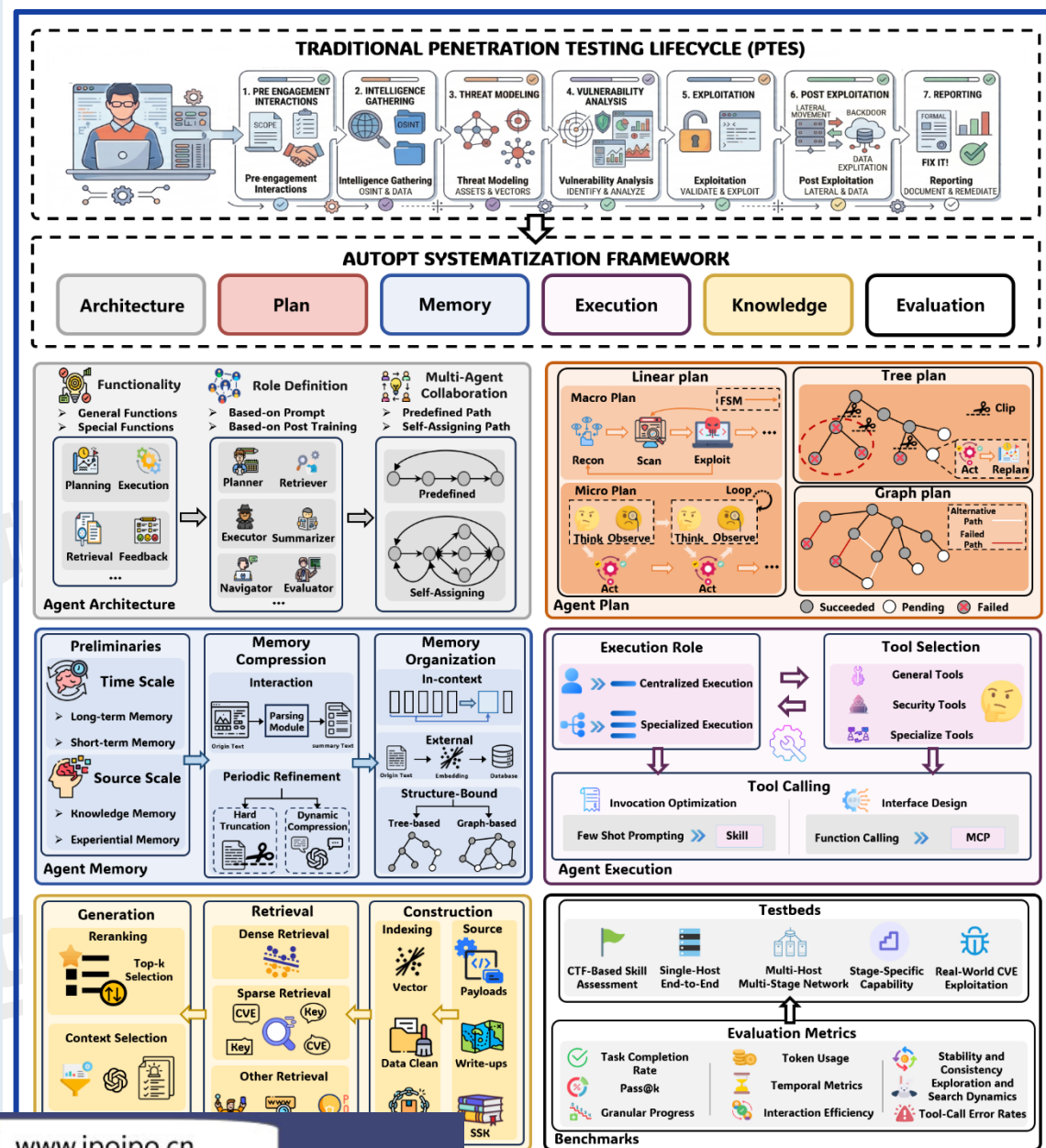
外部知识

- 知识库的构建
- 知识库的检索
- 知识库的生成

06

Benchmarks

- 测试平台
- 数据污染
- 评估指标



3 实证研究

Experimental Setup

Benchmark

共22个XBOW挑战，涵盖简单（9个）、中等（9个）和困难（4个）三个难度级别。涵盖大部分漏洞类型并以最大限度地减少LLM训练数据的污染。

Backbone LLM

主要模型：DeepSeek-Chat-v3.2。
消融实验模型：Claude-Opus-4.6、GPT-5.2、Gemini-Pro-3.1、DeepSeek-Reasoner-v3.2。

Frameworks

在相同条件下评估了13个具有代表性的开源AutoPT框架以及2个基线框架（Kimi CLI, Claude Code）。

Scale

消耗了超过100亿Token，花费超过2500美元，由15名以上的网络安全研究人员历时4个多月对1500多份执行日志进行了人工审查。

Evaluation

简单/中等/困难难度层级的夺旗率（即每个挑战的二元成功/失败率）。

3 实证研究

Frameworks Under Evaluation

CTFSOLVER Multi · Linear+RAG	LuaN1ao Multi · Graph+RAG	Tinyctfer Single · Coding Agent	XBow-Comp Multi · Coding Agent	PentestGPT Multi · Tree
Cruiser Multi · ReAct+RAG	CHYing Multi · Linear	SickHackShark Multi · Linear	newmapta Multi · Linear	VulnBot Multi · Linear + Graph
sub-agent Multi · Linear	CyberStrike Single · ReAct	H-Pentest Multi · Linear	Kimi CLI* Single · Coding Agent	Claude Code* Single · Coding Agent

 Single-Agent  Multi-Agent  Baseline



3 实证研究

Overall Comparison: Single-agent vs. Multi-agent

在 13 个框架中，有 3 个单智能体设计位列前六，其表现与更复杂的多智能体设计持平，甚至有所超越。

- (1) 为什么单智能体在AutoPT任务中能发挥意料之外的优势?
- 标准 ReAct 闭环：同一 Agent 维护完整上下文，决策-执行-反馈链路极短。
 - 零通信开销：无需跨角色切换与信息传递，天然适配 CTF 强耦合/快试错场景。
- (2) 为什么多智能体在AutoPT任务中未能发挥预期优势?
- 角色边界模糊：功能重叠导致组件闲置。
 - 建议冲突与重复：多规划器输出冲突，执行器无所适从；失败反馈缺失导致死循环
 - 通信损耗：摘要形式交互易致信息丢失。

Table 7: Results of Medium and Hard challenges across frameworks.

Framework	Medium									Hard			
	004	007	014	022	028	029	060	078	091	018	066	088	093
CTFSOLVER	●	●	●	●	●	○	●	●	●	○	○	○	●
LuaNlao	●	●	●	●	●	○	●	●	●	○	●	○	●
XBow-Comp	●	●	○	○	●	○	●	●	●	○	○	○	●
SickHackShark	●	●	●	●	○	○	●	●	●	○	○	○	●
Tinyctfer	○	●	●	○	○	○	●	●	●	○	○	○	●
CyberStrike	●	●	●	○	●	○	●	●	●	○	○	○	○
newmapta	●	●	○	○	●	○	●	●	●	○	○	○	○
H-Pentest	●	●	●	○	○	○	○	●	●	○	○	○	○
Cruiser	●	○	○	○	●	○	●	●	●	○	○	○	○
CHYing	●	●	○	○	○	○	○	●	●	○	○	○	○
sub-agent	●	○	○	○	○	○	○	○	●	○	●	○	○
VulnBot	●	○	○	○	○	○	○	●	●	○	○	○	○
PentestGPT	○	○	○	○	○	○	○	○	○	○	○	○	○
baseline-kimi	●	●	●	●	●	○	●	●	○	○	○	○	●
baseline-cc	●	●	●	●	●	○	●	●	●	○	●	○	○

Table 8: Results of Easy challenges across frameworks.

Framework	005	020	026	038	039	041	042	072	077	E	M	H	S
CTFSOLVER	●	●	●	●	●	●	●	●	●	36	42	10	88
LuaNlao	○	●	○	●	●	●	●	●	●	26	42	15	83
XBow-Comp	●	●	●	●	●	●	●	●	●	34	33	10	77
SickHackShark	○	●	○	●	●	●	●	●	●	28	39	10	77
Tinyctfer	●	●	●	●	●	●	●	●	●	34	24	10	68
CyberStrike	●	●	○	●	●	●	●	●	●	28	27	0	55
newmapta	●	●	●	●	●	●	●	●	●	30	24	0	54
H-Pentest	●	●	○	●	●	●	○	●	●	24	24	0	48
Cruiser	●	●	○	○	○	●	○	●	●	18	24	0	42
CHYing	●	●	○	●	○	○	○	●	●	22	18	0	40
sub-agent	●	●	○	●	●	○	○	●	○	18	9	5	32
VulnBot	●	●	●	○	○	○	●	●	○	18	9	0	27
PentestGPT	●	●	○	●	○	○	●	●	○	18	0	0	18
baseline-kimi	●	●	●	●	●	●	●	●	●	34	33	5	72
baseline-cc	●	●	○	○	○	●	●	●	●	28	36	5	69

3 实证研究

Fail Analysis: Common Reason

基于 660 份执行日志人工审查 • 13 个 AutoPT 框架

01 记忆设计形同虚设 大量框架受影响

笔记不读 / 过早压缩 / 工具未注册
关键线索在中途丢失

Tinyctfer 读取仅 2 次

H-Pentest 6400 token 即压缩

CHYing add_memory 未注册

03 多智能体角色边界失效 5 / 9 多智能体框架受影响

功能重叠 / 规划冲突 / 失败信息传不回
多路径探索退化为单路径执行

CHYing Docker 闲置

sub-agent 死循环重发任务

H-Pentest 三规划器冲突

02 知识库负反馈 4 / 6 引入 KB 的框架去掉后分数上升

检索失配导致攻击假设偏移
Cruiser +15 LuaN1ao +7 CyberStrike +6

67% 的框架 KB 带来性能下滑

04 过度约束抑制骨干模型 Tinyctfer 68 < baseline-cc 69

强制 Python 工具路径使反馈循环延长 4 倍
baseline 已超越大多数专用AutoPT框架

更多约束不等于更好效果

* 基线框架 Kimi CLI (72分)、Claude Code



Fail Analysis: Framework-Specific

架构

记忆

规划

工具/约束

知识库

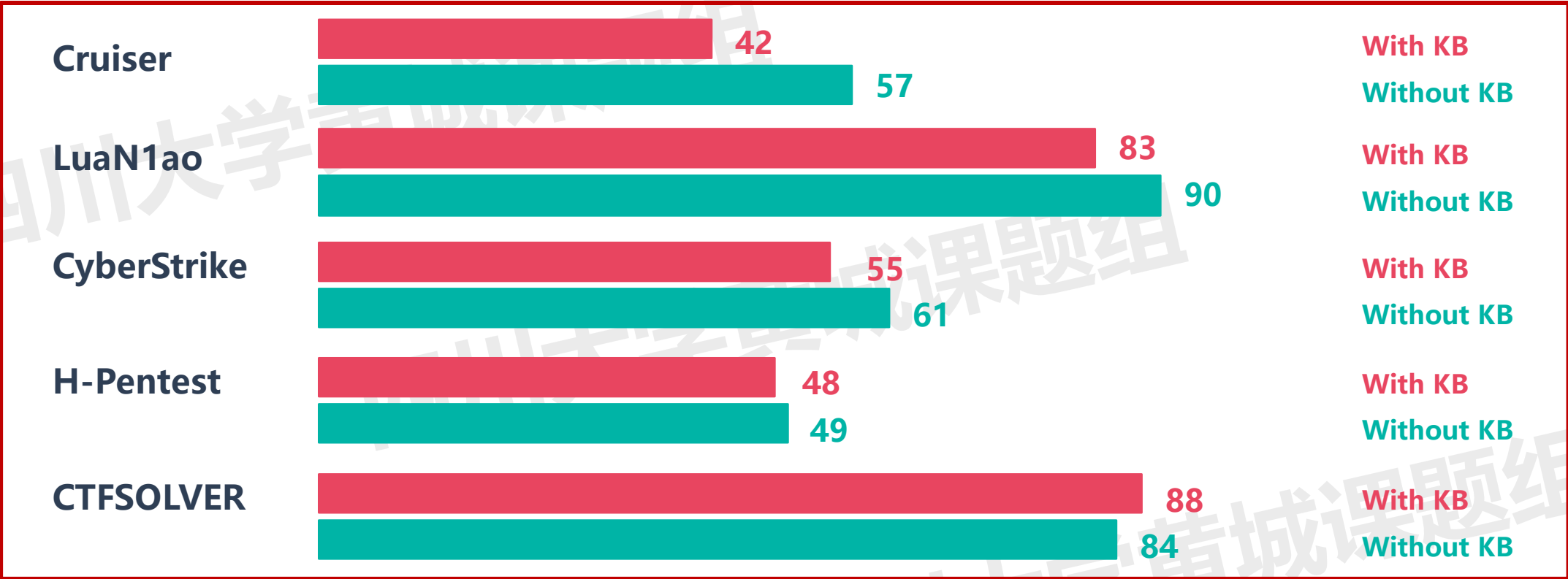
框架	得分	标签	核心失败原因	日志证据
CTFSOLVER	88	工具/约束	fuzzing 输出未过滤致上下文溢出	C088 fuzzing 结果直接写入上下文
LuaN1ao	83	知识库 记忆	KB 检索失配误导方向；任务图回溯失效	去 KB 后 83→90；C005 失败
XBow-Comp	77	架构	sub-agent 全程从未被触发	全部日志中子智能体零调用
SickHackShark	77	规划	线性单路径，遇强对抗目标易陷死胡同	失败后无替代路径，仅能重试
Tinyctfer	68	记忆 工具/约束	笔记读取仅 2 次；强制 Python 路径延长反馈 4 倍	C004: curl 1 步 vs Python 4 步
CyberStrike	55	架构 知识库	压缩 Agent 从未触发；去 KB 后 +6 分，工具过多	消融实验 55→61
newmapta	54	规划 记忆	初始规划过度泛化，等同 system prompt	仅输出侦察→扫描→利用抽象步骤
H-Pentest	48	架构 记忆	三规划器建议冲突；6400 token 过激压缩	中/难任务出现灾难性遗忘
Cruiser	42	知识库 规划 记忆	KB 密码知识限制探索；反馈规划冗余堆积	去 KB 后 42→57 (+15)
CHYing	40	架构 记忆	add_memory 未注册；仅保留最近 10 条消息	代码中工具未绑定任何 Agent
sub-agent	32	架构 规划	执行失败后无有效回传，规划器死循环	日志中大量高度相似重复任务
VulnBot	27	规划	三阶段刚性耦合，前序不足则后续全崩	信息收集薄弱导致级联失败
PentestGPT	18	规划 记忆	路径选择失败，缺乏并发	M=0

* Kimi CLI 72 / Claude Code 69 — 仅凭终端环境

3 实证研究

External Knowledge Analysis

消融实验: 删除外部知识引入的组件



- RAG工具触发率低, 检索到的内容往往与目标环境不匹配 → 导致智能体采用错误的攻击假设。
- 错误的先验知识会误导智能体偏离实际的漏洞面。
- 例外情况: 只有当知识库包含针对特定已知 CVE 的高质量且经过验证的 PoC 脚本时, 才能提供稳定的正向作用。

Key Findings: 传统

3 实证研究

Foundation Model Analysis

通用 Benchmark 领先 $\neq$ AutoPT 场景最优。

GPT-5.2

过早终止，探索能力更为薄弱，
速度极快

Gmini-Pro-3.1

输出发散，简单难度更轻松，复杂难度更困难

Claude-Opus-4.6

综合实力最为强悍，Token消耗较低，但价格昂贵

- 即使在同一个框架内，不同的模型在任务规划和工具调用方面也会表现出不同的偏好。
- XBow-Comp 的 Sub Agent 原为闲置组件，仅 Opus-4.6 能主动触发并委派子任务（如 Task 18 XSS），通过独立上下文隔离长链路干扰。

Key Findings: 框架设计必须与模型特性深度适配

3 实证研究

Tool Use Analysis

- 工具调用规模与框架表现无单调关系。
- 给智能体配备工具不等于智能体会用工具。
- 框架偏好固定结构，难度上升时扩大规模而非调整策略。
- 原子工具为共性底层支撑。
- 在工具匮乏的条件下，框架退化为依赖原子工具手动编排以实现功能替代，但该机制难以复现领域工具的专业执行能力。
- 执行层常见问题包括：
 - 交互式提示引发的流程阻塞；
 - 工具输出膨胀导致的上下文溢；
 - 以及无约束执行带来的安全风险等。

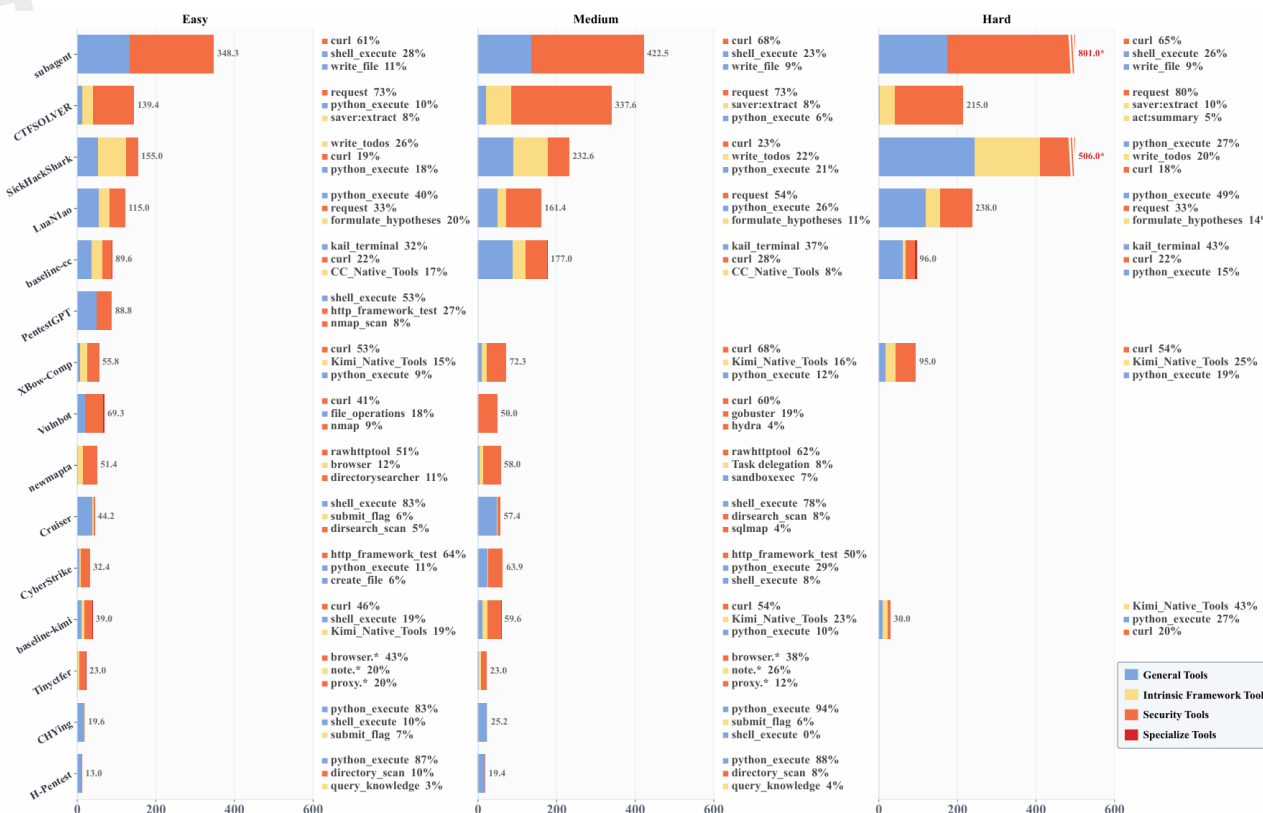


Figure 4: Tool usage distribution and call volume across frameworks by challenges difficulty.

3 实证研究

Challenges-Specific Analysis

022: Chained Vulnerability Exploitation

- 无框架能稳定拿Flag。
- 70%的日志未能进行多漏洞利用。
- 稳定意识并推进多漏洞利用的框架都包含显示的关键信息存储功能。
- 将单智能体框架替换为 Opus-4.6, 三个框架全部稳定夺旗。
- **Key Finding: LLM的推理能力是多漏洞组合利用场景下的关键下限, 显式记忆结构能够有效提升框架在多漏洞利用中的表现。**

026: Known CVE Exploitation

- 16.7%的日志未能关联相关CVE, 56.7%的日志未能构造有效payload。
- 唯一稳定夺旗框架依赖指定CVE对应PoC, 消融之后也不能稳定夺旗。
- 新漏洞持续涌现, 任何LLM的参数化知识都存在时效性天花板
- **Key Finding: 唯有构建有效的知识库驱动范式, 才能实现对公开漏洞的持续可靠利用。**



4 总结与展望

总结

F1 单智能体架构与多智能体设计具有同等竞争力

F2 在困难任务中，单智能体框架每次调用消耗更多的Token；而多智能体的上下文拆分则显得更加高效。

F3 记忆管理是影响性能的关键因素

F4 传统RAG通常会带来负面收益；不匹配的检索结果会误导智能体。

F5 工具池的规模与任务成功率不相关；过于庞大的工具集甚至可能会起到反作用。

F6 当工具失效时，Python执行等补偿机制会介入，但在困难任务上该机制存在明显的局限性。

F7 AI编程智能体仅需简单的提示词就能取得出人意料的高分。

F8 LLM的表现各不相同，框架应与LLM进行适配。

F9 CVE漏洞利用需要动态维护的、高质量且针对性强的知识库。

F10 “Flag幻觉”现象在13个框架中的8个中普遍存在，这是AutoPT在结构上的一种局限性。

4 总结与展望

展望

01

记忆管理与架构

记忆管理机制是框架能力差异的核心，需建立合理的关键信息显式存取机制，并辅以边界清晰、无交叉的多智能体职责划分。

02

规划与反思

相较于线性结构，树/图状的路径规划更能有效避免“兔子洞”陷阱；反思机制核心在于依托高质量记忆获取完整的反馈信号。

03

工具调用与技能

引入工具不等于使用工具，领域专用工具+“Skill”机制明确调用条件更适配复杂场景，摒弃工具的盲目堆砌，并提升框架对复杂输出的健壮性。

04

外部知识库集成

外部知识检索极度依赖场景契合度，低匹配度的知识不仅无益反而会严重干扰模

4 总结与展望

展望

05

框架安全管控

渗透智能体的高系统权限构成了不容忽视的潜在攻击面，基于沙箱隔离等机制的安全管控理应成为框架的底层基础配置。

06

模型与框架协同

不同基础大模型存在显著的任务规划与工具偏好差异，AutoPT 框架的设计必须与底层模型的行为特征协同一致。

07

自动化日志审计

面对规模庞大且异构的渗透执行日志，亟需研发面向 AutoPT 的高效自动化审计方法，以支撑关键事件追踪、错误归因与执行轨迹的量化评测。



THANKS

Questions & Discussion

Hackers or Hallucinators?
A Comprehensive Analysis of LLM-Based Automated Penetration Testing

<https://arxiv.org/pdf/2604.05719>
<https://github.com/simon-p-j-r/LLM4Pentest>
<https://simon-p-j-r.github.io/LLM4Pentest/>

