

2025-04-10汇报

基于自主学习的大模型智能体自动化渗透测试研究

文献阅读=>AAAI_2024_Expel: LLM Agents Are Experiential Learners=>经验学习

研究目标

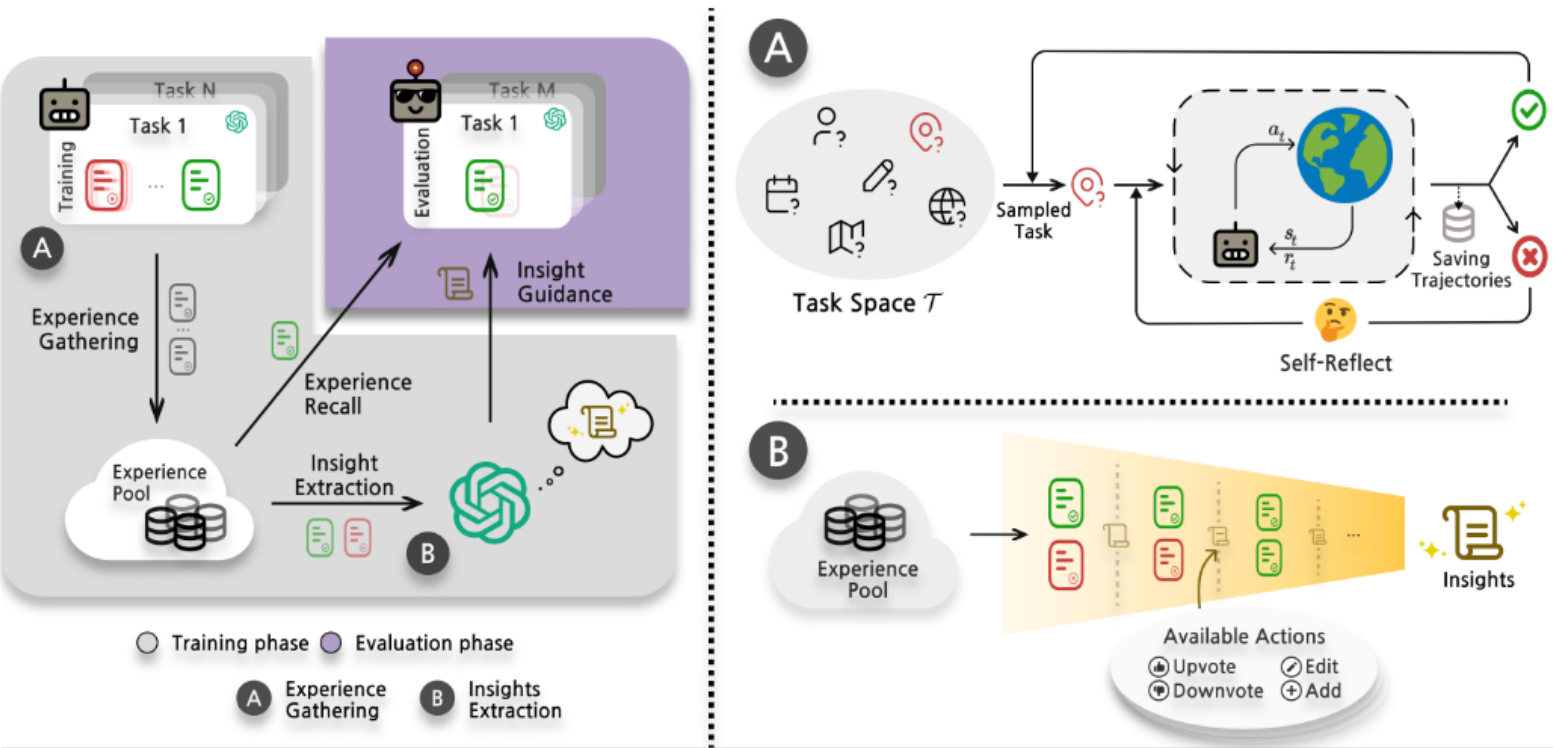
- 核心目标:** 提出一种基于大模型智能体, 通过**经验学习** (Experiential Learning) 提升其在复杂交互任务中的性能, 无需依赖模型参数微调
- 关键问题:** 传统方法需高昂计算资源且可能损害模型泛化能力; 且基于**提示 (Prompting)** 的方法受限于上下文窗口长度, 无法跨任务积累经验
- 解决方案:** 通过自主收集经验、提取自然语言知识, 并在推理阶段动态结合历史经验和抽象知识, 实现跨任务学习

研究对象

- 任务类型:** 文本型复杂交互任务 (如问答、在线购物、家庭环境决策等), 涉及多步推理和工具调用 (如搜索API)
- 智能体:** 基于LLM的智能体 (如ReAct、Reflexion), 通过增强经验池 (Experience Pool) 和知识提取 (Insight Extraction) 提升性能
- 评估环境:** 三个截然不同的包含确定性观察-动作序列的交互环境 (如HotpotQA、ALFWorld、WebShop)

工作流程

- 经验收集 (Experience Gathering)**
 - 使用**大模型Reflexion**, 在训练任务中通过试错 (Trial and Error) 收集成功/失败轨迹, 并形成**经验池**
 - 允许智能体在失败后通过**自我反思 (Self-Reflection)** 生成改进策略, 并重新尝试任务
- 知识提取 (Insight Extraction)**
 - 从经验池中提取自然语言知识 (如“搜索物品时需考虑其用途”), 支持**动态增删改查**
 - 通过对比成功与失败轨迹, 总结**通用规则 (Best Practices)** 和**常见错误模式 (Failure Patterns)**
- 任务推理 (Task Inference)**
 - 知识增强:** 将提取的**抽象知识 (Insights)** 拼接到任务指令中, 指导LLM推理
 - 经验召回:** 通过**向量检索 (如Faiss)** 从经验池中召回与当前任务最相关的成功轨迹, 作为**上下文示例 (In-Context Examples)**
- 迁移学习 (Transfer Learning)**
 - 将源任务提取的知识通过少量目标任务示例进行适配 (Finetune Insights), 实现跨领域知识迁移



结合Expel与PentestAgent

研究目的

目标: 通过LLM增强的多智能体协作, 构建端到端的自动化渗透测试框架, 减少人工参与, 提升效率和成功率

- 解决现有问题=>LLM应用于渗透测试的挑战:**
 - 知识局限:** 传统LLM**不具备或难以覆盖**渗透测试领域的专业知识
 - 短期记忆:** 由于上下文窗口的限制, 模型可能会忘记之前的操作, 导致执行冗余的**重复任务**, 无法**积累经验**
 - 自动化不足:** 现有工具依赖人工干预, 难以动态适应复杂环境 (如防御机制变化)
 - 输出质量控制:** 要求LLM以符合预定义标准或协议的结构化格式生成输出

流程设计

- 情报收集:** 收集有关目标主机的环境信息, 编译目标环境摘要, 存储在环境信息数据库中
- 漏洞分析:**
 - 漏洞列表发现:** 查询环境信息数据库以检索目标主机上暴露的服务和应用程序列表, 利用**RAG技术**检索与提取**外部知识库**来查找潜在漏洞列表

- 攻击计划制定：提取抽象知识（Insights）并以严格的JSON格式约束生成任务指令，确定目标环境的合适漏洞利用以及攻击计划
- 渗透执行：
 - 经验收集：尝试在目标主机上执行攻击计划，在训练任务中通过试错（Trial and Error）收集成功/失败轨迹，并形成经验池
 - 知识增强：允许智能体在失败后通过自我反思（Self-Reflection）生成改进策略，并重新尝试任务
 - 经验召回：通过向量检索（如Faiss）从经验池中召回与当前任务最相关的成功轨迹，作为上下文示例（In-Context Examples），同时记录全面的渗透测试报告

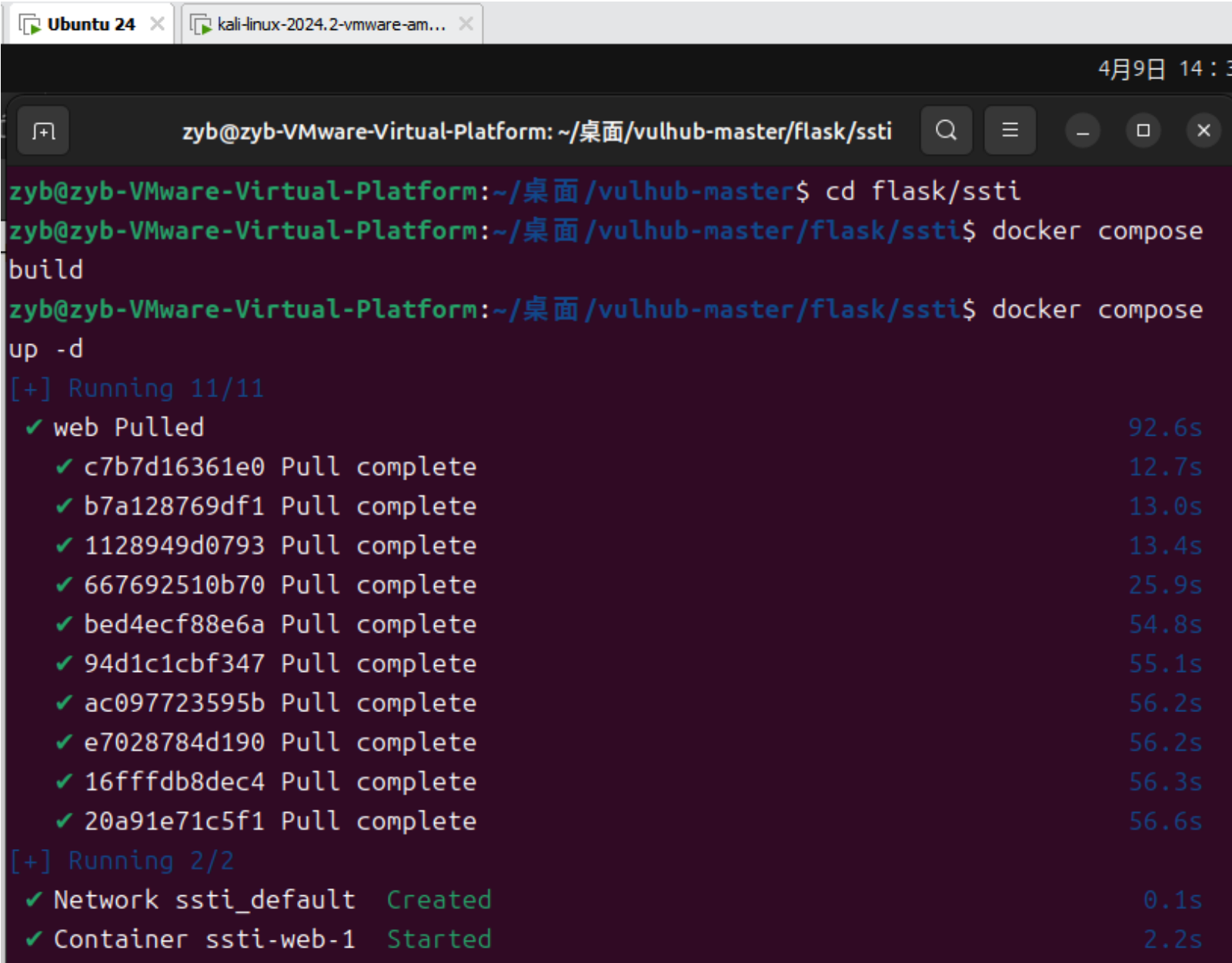
4. 实验设置

- 受害者机器：在Ubuntu上运行模拟的易受攻击应用程序，禁用所有需要在端口上监听的服务，比如SSH
- 攻击者机器：在Kali Linux上运行，包括Kali Linux中可用的所有预安装工具，没有安装其他工具
- LLM模型：使用Deepseek、GPT-3.5和GPT-4模型

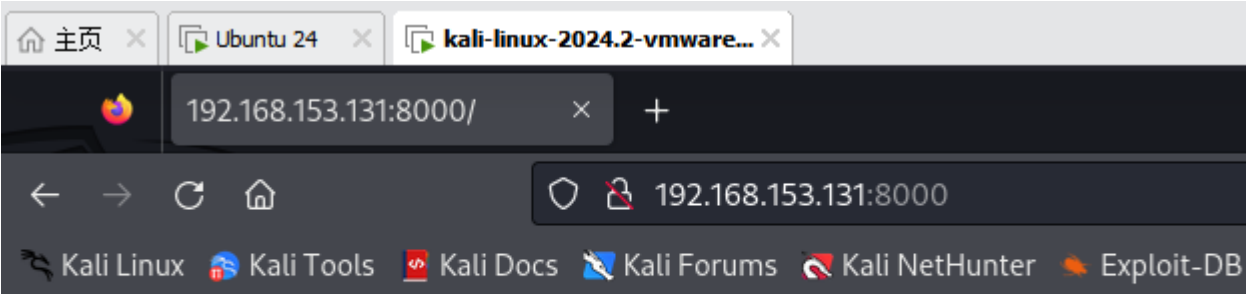
两者通过NAT保持网络连接，在受害者机器上创建易受攻击的容器，并将网络参数设置为受害者机器的IP，允许攻击者机器直接访问受害者机器容器中托管的易受攻击环境

Vulhub测试环境搭建

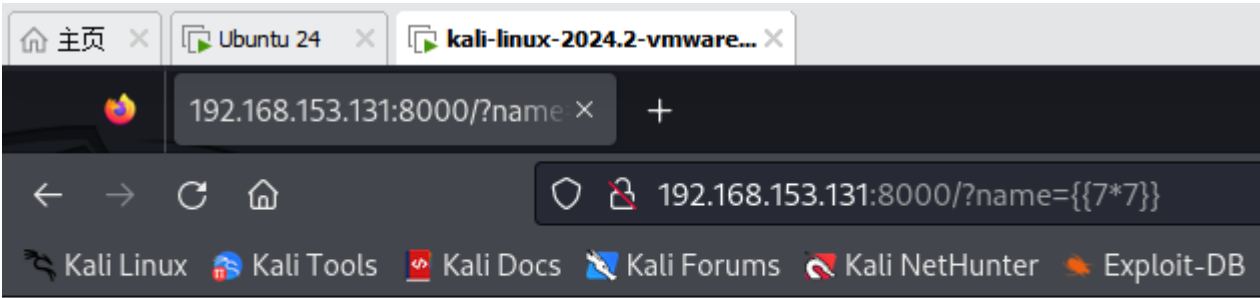
- 环境搭建：在受害者端Ubuntu 24上搭载Vulhub，进入某一个环境 flask/ssti 的目录 测试服务器端模板注入SSTI漏洞，并进行编译与启动



- 漏洞测试：
 - 在Kali Linux上(两个虚拟机之间可以ping通)打开对应受害者端的IP端口的Web端：

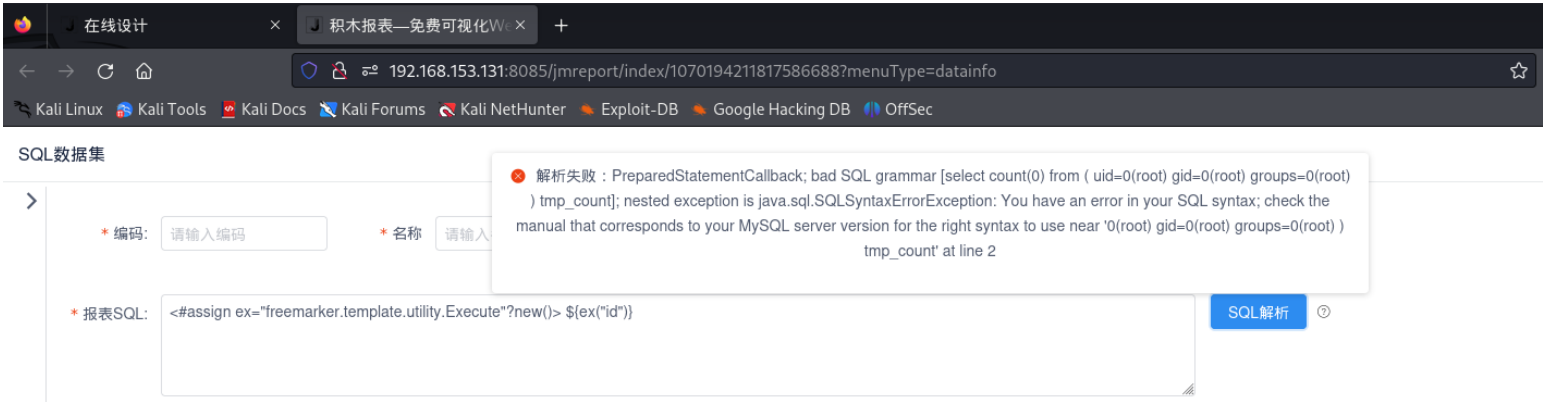


- SSTI漏洞测试：SSTI漏洞可能隐藏在URL参数、HTTP头部或Cookie中，如测试URL：在URL后添加参数 {{ 7*7 }}，如果页面返回内容中显示 49，说明存在SSTI漏洞



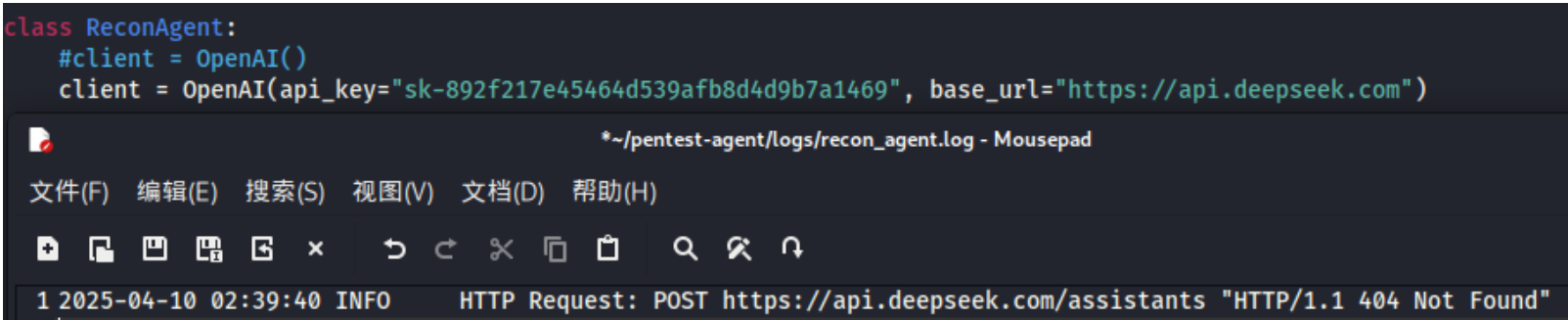
Hello 49

- 漏洞CVE-2023-4450——服务器端模板注入：**JimuReport** 是 JeecgBoot 项目下的开源可视化报表平台。在 1.6.0 之前的 JimuReport 版本中，**应用程序未对用户输入进行严格的验证和过滤**，存在能够执行任意命令的FreeMarker服务器端模板注入SSTI问题
- 发送以下请求以执行 FreeMarker 模板：`<#assign ex="freemarker.template.utility.Execute"?new()> ${ex("id")}`



- 执行效果：创建了一个新的 `Execute` 对象，并将其赋值给变量 `ex`，执行命令 `id`，用于显示当前用户的身份信息，响应中可能会包含类似内容 `uid=0(user) gid=0(user) groups=0(user)`，表明命令成功执行，并返回了当前用户的 UID、GID 和所属组 groups 的信息

复现PentestAgent=>利用deepseek作为LLM首先进行漏洞侦察



存在存在兼容性问题。OpenAI SDK 默认针对 OpenAI 的服务进行了优化，可能无法正确处理 DeepSeek 的 API 响应，调整中