2025-05-15汇报——思路

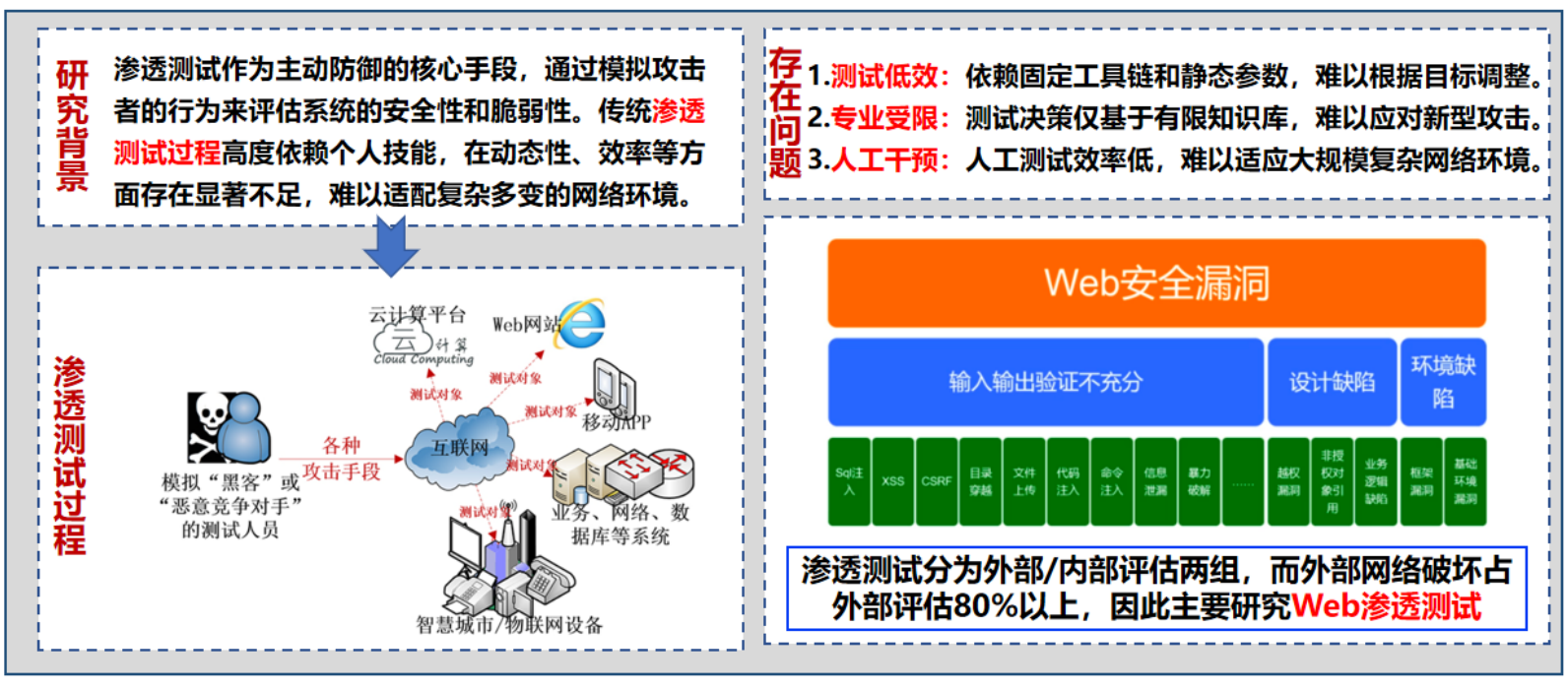
基于大模型多智能体的自动渗透测试系统

引言

网络安全在数字化时代的重要性日益凸显，渗透测试作为主动防御的核心手段，通过模拟攻击者的行为来评估系统的安全性和脆弱性。传统上，渗透测试主要依赖安全专家的经验，采**用人工方式**逐步探测目标系统的漏洞，并评估其潜在风险。然而，这种方法存在显著的局限性：测试过程`高度依赖个人技能`，导致测试结果的可重复性和一致性较差；面对大规模复杂网络环境时，`人工测试效率低下`，难以实现全面覆盖；此外，随着新型攻击技术的不断演进，传统渗透测试方法在动态`适应新环境`方面表现出明显的滞后性。

为提升效率，**自动化渗透测试技术**应运而生。当前主流方案通常基于预定义的漏洞扫描规则、渗透测试框架和自动化攻击脚本，能够在无人干预的情况下执行部分测试任务。然而，这类系统仍然存在诸多不足：其决策逻辑大多基于静态规则或有限的知识库，`难以适应动态变化`的网络环境；在复杂攻击场景下，自动化系统缺乏灵活的推理能力，无法像人类专家一样调整测试策略；此外，现有方法在绕过防御机制等方面仍显不足，导致测试深度和有效性受限。这些局限性表明，当前的自动化渗透测试仍未能完全摆脱对人工干预的依赖，亟需更先进的智能决策机制来提升其自主性和适应性。

近年来，**大语言模型LLM**的快速发展为自动化渗透测试提供了新的可能性。以GPT-4为代表的大模型展现出强大的自然语言理解、代码生成和逻辑推理能力，使其能够理解渗透测试任务的需求并生成相应的测试策略。在网络安全领域，大模型已初步应用于漏洞分析、攻击模式识别和恶意代码检测等任务，表现出超越传统自动化方法的潜力。更重要的是，多智能体系统与大模型的结合，可以构建分工协作的渗透测试智能体集群，例如情报收集、计划制定、攻击执行等不同功能的智能体协同工作。这种架构能够更灵活地适应不同测试场景，并在复杂网络环境中实现更高效的渗透测试。



尽管大模型为自动化渗透测试带来了新的机遇，但其实际应用仍面临以下诸多挑战:

1)**决策逻辑固化**：测试过程通常依赖固定的工具链以及静态参数配置，缺乏对目标环境实时状态的感知能力，难以根据网络拓扑变化、防御机制调整或入侵检测系统的响应情况权衡测试效率与隐蔽性；

2)**专业知识受限**：当前大模型在漏洞机理理解、攻击面分析等专业领域存在显著的知识盲区。同时，大模型上下文窗口有限，缺乏持续学习能力，无法从历史测试案例中提取经验知识，应对动态变化的网络环境；

3)**记忆管理问题**：由于缺乏统一的工作记忆管理机制，各智能体仅关注自身子任务，形成的异构记忆片段无法有效整合，导致多智能体间的工作记忆状态呈碎片化，破坏渗透过测试流程连贯性与一致性。

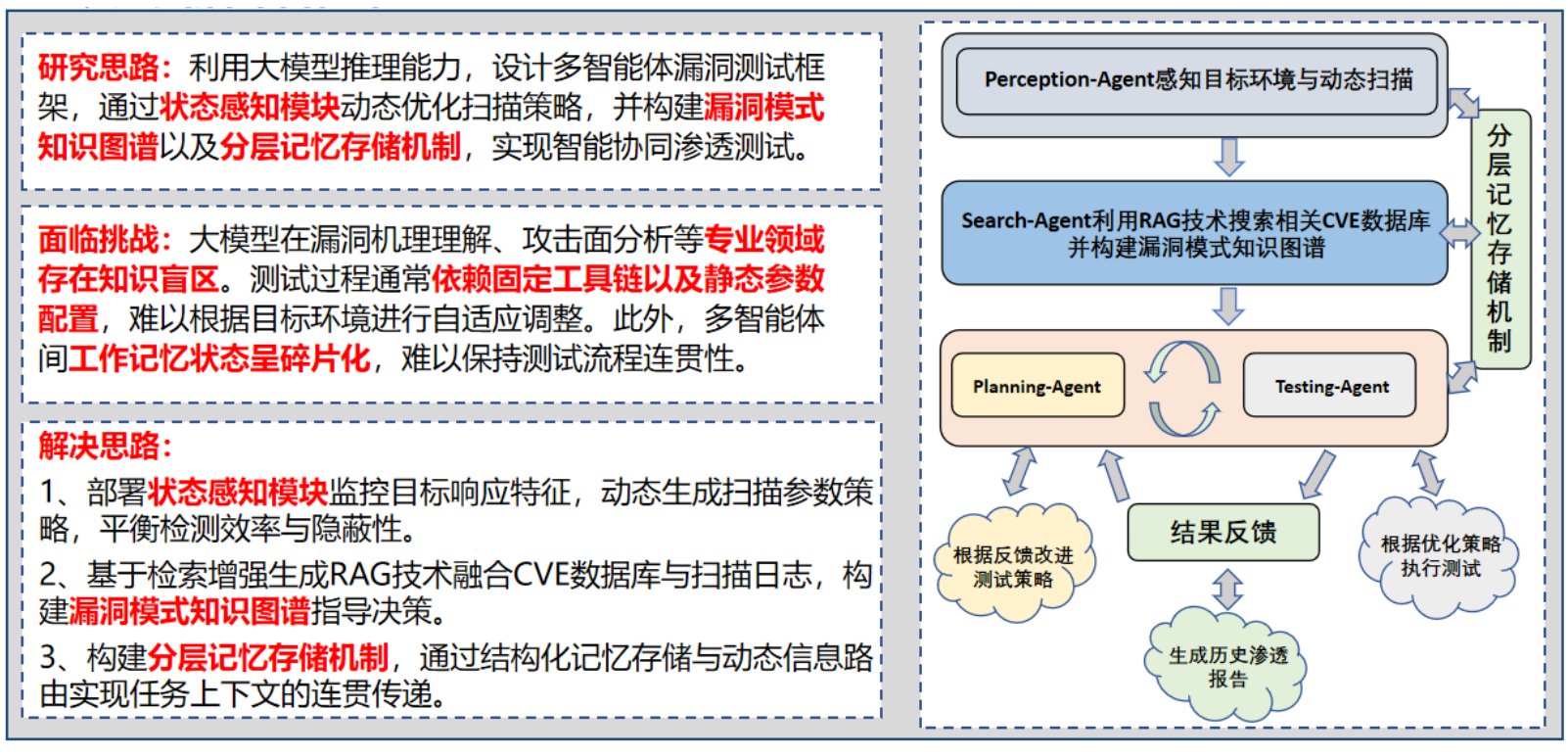
为了克服这些挑战，本文提出一种新的基于大模型的自动化渗透测试框架**CogniPentest**。该框架旨在利用大模型推理能力，不断集成新技术并结合历史经验增强渗透测试知识库，有效应对动态变化的网络环境。我们的目标是提高自动化渗透测试的持续学习能力，能够根据目标环境动态调整渗透测试策略。

CogniPentest采用多智能体设计，主要由四个部分组成：Perception-Agent、Search-Agent、Planning-Agent、Testing-Agent。这些智能体涵盖了自动化渗透测试的三个主要阶段：目标侦察、漏洞分析与测试执行，在渗透测试过程中各自承担特定任务的责任。在各智能体之间构建**分层记忆存储机制**，通过结构化记忆存储与动态信息路由实现任务上下文的连贯传递。

在目标侦察阶段，Perception-Agent根据目标主机IP，通过**状态感知模块**监控目标响应特征，动态调用针对性扫描工具以及生成扫描参数，执行侦察命令并收集目标主机的全面信息。简要分析侦察信息后，存储在环境信息数据库中以供进一步参考。

在漏洞分析阶段，Search-Agent在环境信息数据库中，查询主机使用的具体服务和应用程序后，识别潜在的攻击面，同时使用检索增强生成RAG技术，检索相关CVE数据库、历史渗透报告等内容并**构建漏洞模式知识图谱。**

在测试执行阶段，Planning-Agent根据目标环境信息以及知识图谱内容制定合适的渗透测试策略。而Testing-Agent尝试在目标主机上执行计划好的测试指令，并将测试结果反馈至Planning-Agent以**优化渗透测试策略**，并进一步执行测试。最后记录全面的渗透测试报告，对应生成渗透测试模板供以后参考。



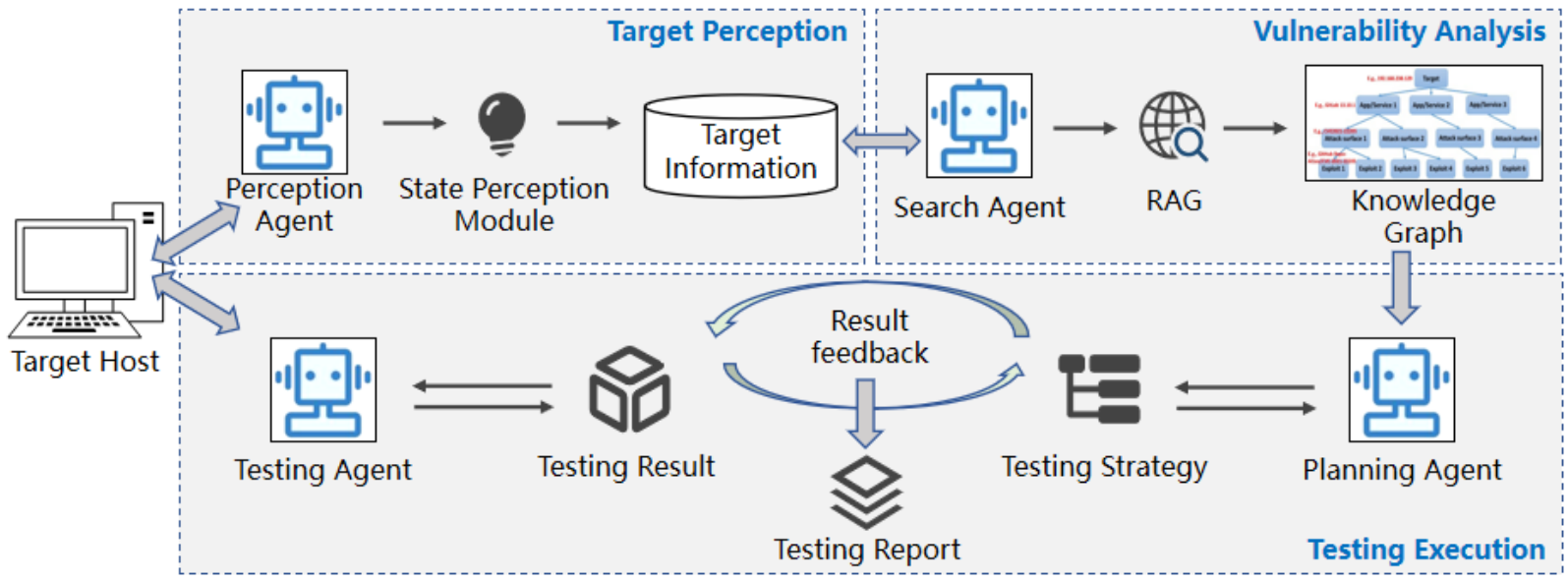
这种全面的方法有望减轻对人工干预的依赖，增强自动化渗透测试系统的持续学习能力。综上所述，我们做出了以下贡献：

设计了**CogniPentest**，一种基于大模型多智能体的自动渗透测试系统，集成了检索增强生成RAG技术构建漏洞模式知识图谱与分层记忆存储机制，以增强渗透测试系统的可持续学习能力以及测试过程连贯性。

系统设计

系统概述

CogniPentest由四个主要部分组成Perception-Agent、Search-Agent、Planning-Agent、Testing-Agent。这些智能体写作执行自动化渗透测试的三个主要阶段：目标侦察、漏洞分析与测试执行。



目标侦察

目标侦察阶段分为三个具体步骤：初始目标扫描、具体对象扫描、深度覆盖扫描。在初始目标扫描阶段，Perception-Agent感知智能体主要解析目标IP端口与服务版本，并以严格的JSON格式输出服务响应延迟、IDS告警频率等反馈情况与技术栈清单，为后续阶段提供精准输入。在具体对象扫描阶段，Perception-Agent感知智能体根据反馈情况，通过**状态感知模块**构建环境状态向量`S = {服务及版本, 防御强度, 漏洞暴露面}`，并根据目标服务匹配针对性的检测工具以及扫描参数，输出调整指令进行具体对象扫描。为突破传统扫描的浅层信息采集，Perception-Agent感知智能体还需要进行深度覆盖扫描，该过程也需要动态选择深度扫描工具与参数，平衡覆盖度与隐蔽性，具体如下：

- **状态感知与工具匹配**：
  - **防御等级评估**：

```
# 防御等级计算（基于告警频率、防火墙规则复杂度等）
defense_level = calculate_defense_level(scan_logs)
```

  - **动态工具选择**：

```
if defense_level < 0.5:
    tools = ["nessus", "openvas"]  # 启用全面扫描
elif defense_level >= 0.5:
    tools = ["nmap --script vuln", "nikto -C all"]  # 启用隐蔽扫描
```

- **参数优化**：
  - **场景1：低防御等级（`defense_level < 0.5`）**：

```
nessus --policy "Full Scan" --target 192.168.1.100  # 全面扫描
```

  - **场景2：高防御等级（`defense_level >= 0.5`）**：

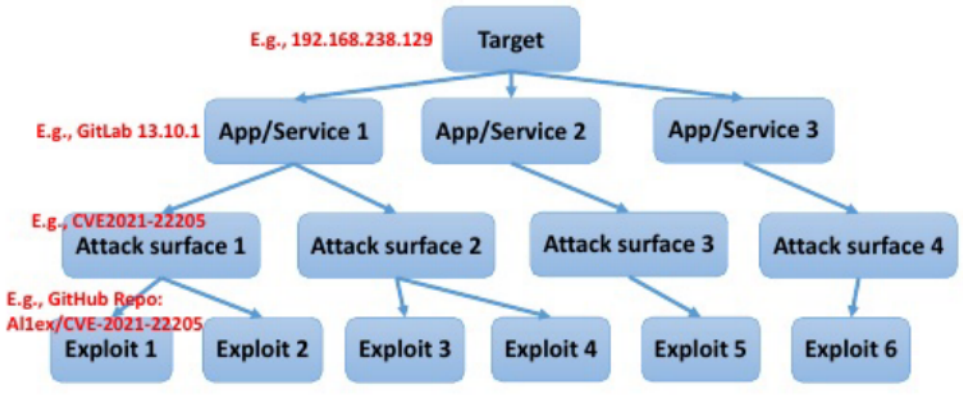
```
nmap -T2 --script vuln --scan-delay 5s 192.168.1.100  # 低速隐蔽扫描
```

目标侦察阶段结束后，将得到的侦察信息存储在环境信息数据库中以供进一步参考。

漏洞分析

在**漏洞分析**阶段，Search-Agent在环境信息数据库中，查询主机使用的具体服务和应用程序后，识别潜在的攻击面，同时使用检索增强生成RAG技术，检索相关CVE数据库、历史渗透报告等内容并**构建漏洞模式知识图谱。**

- **Search-Agent树结构知识图谱构建**：
  - **顶层**：目标主机IP及端口号
  - **初始分类**：基于渗透测试阶段（`侦察→初始访问→服务及版本`）
  - **中间层**：漏洞类型（`注入类、配置错误、权限漏洞、CVE编号`）
  - **叶子层**：具体漏洞实例（`已有测试案例、历史相关渗透报告`）



测试执行

在**测试执行**阶段，首先，Planning-Agent根据目标环境信息以及知识图谱内容制定合适的渗透测试策略。而Testing-Agent尝试在目标主机上执行计划好的测试指令。它从环境信息数据库中检索必要的操作细节，调试执行错误，并将测试结果反馈至计划代理以**优化渗透测试策略**，并进一步执行测试。同时，记录全面的渗透测试报告，对应生成渗透测试模板供以后参考。

- **Planning-Agent生成渗透测试策略**：
  - **策略生成**：
    - 输入：目标环境（如`OS=Windows Server 2022, 开放端口=80/443/3389`）+漏洞知识图谱子图。
    - 大模型制定合适的渗透测试策略，生成**阶段式攻击链**，同时根据目标状态权衡效率与隐蔽性：

```
Phase 1: Web攻击 → CVE-2021-31166 (IIS RCE) → 获取Webshell
Phase 2: 横向移动 → CVE-2020-1472 (Netlogon) → 域控权限获取
```

- **Testing-Agent执行渗透测试指令：**
  - **攻击执行**：
    - 根据Planning-Agent生成的渗透测试策略集成所需工具链，如：SQLMap、Mimikatz。通过大模型生成上下文感知的Exploit代码：

```
# 针对Cloudflare防护的XSS绕过载荷
payload = "<svg/onload=eval(atob('{{base64编码的恶意脚本}}')>"
```

- **测试结果实时反馈——自反思**：
  - 监控执行结果（如HTTP 403响应、IDS告警），触发以下自适应机制：
    - **失败归因分析**：使用决策树模型分析失败根因

```
if "WAF" in last_3_errors:
    root_cause = "载荷特征被识别"
elif "EDR" in last_3_errors:
    root_cause = "内存行为检测触发"
```

    - **优化测试策略**：若当前攻击链失败超过3次，自动切换到备用路径（如从Web攻击转向钓鱼邮件），或反馈回Planning-Agent进一步生成新渗透测试策略。
- **渗透测试报告生成**：
  - 自动生成或更新结构化报告，包含：漏洞对应测试指令报错、测试路径时间线记录