

# 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

### 一、基于大模型智能体的漏洞动态扫描策略优化方法

#### 1. 现有流程核心作用

- 自动化信息收集：通过Nmap快速获取目标主机的端口、服务、版本等基础信息
- 上下文驱动决策：大模型根据扫描结果动态生成后续步骤，替代人工分析决策
- 漏洞定向验证：通过逐步深入的扫描策略收敛到潜在漏洞点

```
(.venv)-(kali㉿kali)-[~/pentest-agent/agents]
$ python3.11 recon_agent.py
```json
{
  "analysis": "None",
  "next_step": "First, perform a basic Nmap scan to identify the operating system and services running on the target host, especially focusing on port 8080.",
  "executable": "nmap -O -sV -p 8080 192.168.153.131"
}
```

```json
{
  "analysis": "The Nmap scan reveals that port 8080 is running Jetty 9.4.37.v20210219, a Java-based web server and servlet container. The target host is likely running a Linux kernel between versions 4.15 and 5.19.",
  "next_step": "Perform a detailed scan of the Jetty service on port 8080 using Nmap scripts to gather more information about the web application, such as HTTP headers, server details, and potential vulnerabilities.",
  "executable": "nmap -p 8080 --script=http-title,http-server-header,http-headers,http-enum 192.168.153.131"
}
```
```

#### 2. 局限性

- 工具链僵化：大多使用固定工具如：Nmap，未动态集成其他工具，且扫描参数固定，如：`-O -sV`，易导致侦察低效或触发防御
- 决策模式静态：大模型建议基于当前扫描结果，无法从历史扫描中积累经验优化后续决策
- 效率瓶颈：
  - 串行执行导致时间成本高，如：先扫描端口再单独检测服务
  - 缺乏多工具协同，如：下图中已经发现 Jetty服务具体版本为9.4.37，存在已知漏洞，并给出的下一步建议包括验证；但侦察代理仍只能继续调用gobuster扫描工具侦察目标服务器上可能存在的隐藏路径或文件；无法协同其他代理执行攻击，难以及时进行漏洞验证

```
```json
{
  "analysis": {
    "8080": {
      "service": "Jetty 9.4.37.v20210219",
      "findings": {
        "default_content": "Present (Example Domain page)",
        "management_interface": "Not found at standard /manager/html location",
        "error_disclosure": "Full Jetty version disclosed in 404 error page",
        "security_concerns": [
          "Jetty 9.4.37 has known vulnerabilities (CVE-2021-28165, CVE-2021-28169)",
          "Default error pages reveal server information"
        ]
      }
    }
  },
  "next_step": "Since standard management interfaces aren't exposed, we should: 1) Check for other common web server paths 2) Perform directory brute-forcing 3) Search for exploits specific to Jetty 9.4.37.v20210219",
  "executable": "gobuster dir -u http://192.168.153.131:8080 -w /usr/share/wordlists/dirb/common.txt"
}
```
```

#### 3.DeepSeek漏洞动态扫描方案

- 漏洞扫描工具链动态编排：突破Nmap依赖，将Nmap替换为模块化工具调用层，支持动态加载各类渗透工具，实现多工具智能调度
  - 工具能力知识库：将渗透工具抽象为可编程接口，定义其输入输出能力

```
{
  "工具名": "nmap",
  "能力": ["端口扫描", "服务识别"],
  "隐蔽性": 0.3,
  "适用场景": ["初期信息收集"]
},
{
  "工具名": "whatweb",
  "能力": ["web指纹识别"],
  "隐蔽性": 0.7,
  "适用场景": ["web服务深度分析"]
}
```

- 动态选择算法：根据目标状态选择最优工具组合，如：
  - 目标状态示例：`{"需隐蔽性": 0.8, "服务类型": "web"}`
  - 形成：`[初步扫描工具]-[具体对象扫描工具]-[深度扫描工具]`
- 上下文感知参数优化：避免固定参数导致的低效或触发防御
  - 参数生成模板：

```
prompt = f'''
目标环境特征：
- 服务：{service_info}
- 防御强度：{defense_level}
请为Nmap生成优化扫描参数，要求：
- 隐蔽性优先（如随机化扫描间隔）
- 合并必要检测项（如OS识别与脚本扫描）
输出JSON格式：{{"command": "nmap ..."}}
'''
```

◦ 执行示例：

- 原命令： `nmap -O -sV -p 8080` → 优化后： `nmap -T3 --scan-delay 1s-5s -sS -A -p 8080`，对比效果：

| 特性     | 原命令 ( <code>nmap -O -sV -p 8080</code> ) | 优化后命令 ( <code>nmap -T3 --scan-delay 1s-5s -sS -A -p 8080</code> ) |
|--------|--|---|
| 扫描速度   | 默认速度（未指定）                                | 中等速度（ <code>-T3</code> ）  |
| 隐蔽性    | 较低（无延迟，可能被检测到）                           | 较高（随机延迟 + SYN 扫描）   |
| 信息收集范围 | 仅包含 OS 和服务版本检测                           | 包含 OS、服务版本、脚本扫描、路由信息等全面信息   |
| 适用场景   | 快速获取基本信息                                 | 需要全面信息且希望避免被检测到的场景  |

- 经验驱动的策略进化：避免重复低效决策，实现持续优化

- 经验池存储：记录历史扫描结果与决策效果

```
class Experience:
    def __init__(self, state, action, reward):
        self.state = state # 扫描前环境状态
        self.action = { # 执行动作
            "工具": "nmap",
            "参数": "-sV -O",
            "目标IP": "192.168.153.131"
        }
        self.reward = reward # 奖励值（漏洞发现数 - 触发告警数）
```

## 二、基于大模型智能体的自动化攻击链生成优化方法

### 1. 现有大模型驱动攻击方法的局限性

- 工具调用僵化：依赖预定义工具链，无法动态适配复杂场景
- 经验不可复用：成功攻击路径未形成结构化知识，重复场景需重复决策
- 自主学习能力缺乏：当遇到新的环境时，难以根据过去经验生成改进策略进行有效应对

### 2.DeepSeek动态生成攻击链方案

- 工具链动态编排引擎：

- 语义化工具能力描述：

```
{
  "sqlmap": {
    "功能": "SQL注入检测与利用",
    "输入要求": {"需已知注入点参数"},
    "输出能力": {"获取数据库名", "提取表数据"},
    "隐蔽性评分": 0.6,
    "环境依赖": ["Python3", "网络可达性"]
  },
  "metasploit": {
    "功能": "漏洞利用与载荷投递",
    "输入要求": {"目标IP", "漏洞CVE编号"},
    "输出能力": {"获取Shell访问", "权限提升"},
    "隐蔽性评分": 0.4,
    "环境依赖": ["Ruby环境", "Payload兼容性"]
  }
}
```

- 上下文感知参数生成：如： `目标状态: {防御强度: 0.8, 需隐蔽性: 0.9, 环境依赖: xxx}`

- 攻击链动态生成与优化机制：

- 动态工具选择：生成定制化攻击指令，并从工具知识库筛选符合要求的工具，执行命令
  - 经验池存储：最多N次的情况下，重复执行训练任务，收集反馈的成功/失败轨迹，形成经验池；同时通过对比成功与失败轨迹，总结通用规则和常见错误模式
  - ExpeL经验学习：允许智能体在失败后，通过自反思(Self-Reflection)机制结合RAG检索技术生成改进策略，并重新尝试任务

