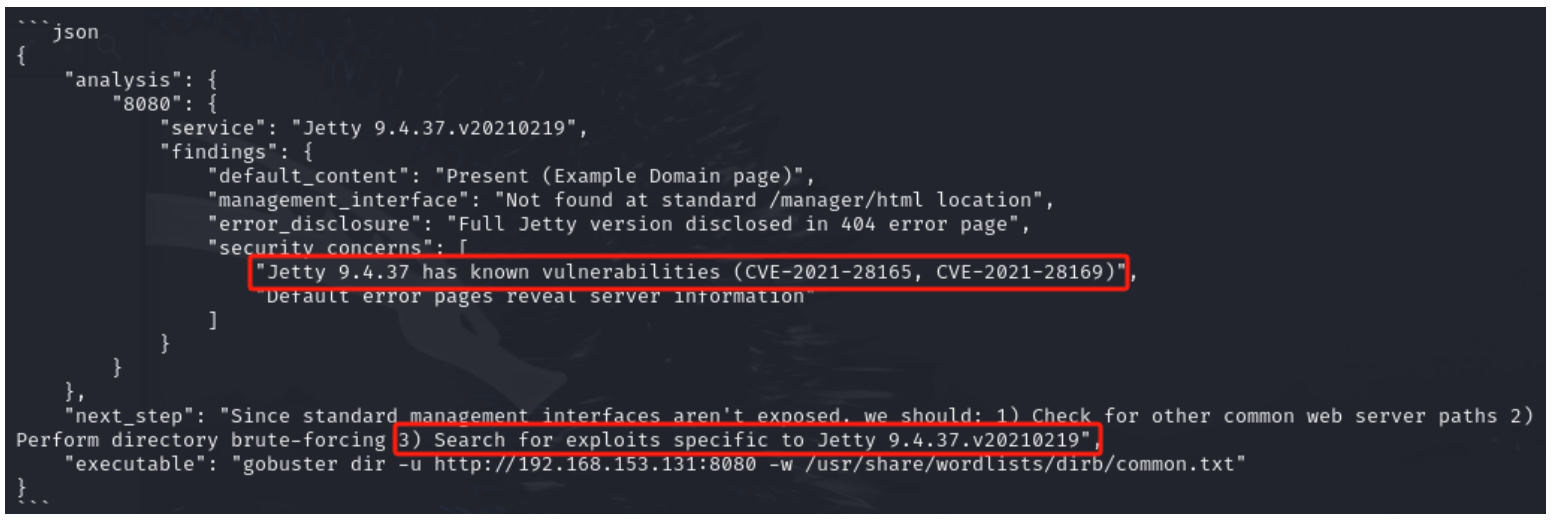
2025-05-08汇报

基于大模型智能体的Web服务自动化漏洞扫描与动态优化方法

现有漏洞扫描局限性

- **工具链僵化**：大多使用初步思考的某几个固定工具如：Nmap，未动态使用其他更合适的工具，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"
}
```

Web服务自动化漏洞扫描与动态优化方案

一、初始阶段：轻量级信息收集与分类

- **目标**
  - **解决问题**：传统渗透测试依赖人工分析目标基础信息，效率低且易遗漏关键服务
  - **创新性**：通过大模型智能解析端口与服务版本，自动标记风险点，为后续阶段提供精准输入
- **实现方法**
  - **工具链编排**：
    - **核心工具**：`Nmap`（服务探测） + `Wappalyzer`（技术栈识别）
    - **执行命令**：

```
nmap -sV -T3 -p 80,443 --script=http-title 192.168.1.100
wappy -u http://192.168.1.100
```

  - **严格输出**：输出JSON格式的风险标签与技术栈清单

```
{
  "service": "Apache/2.4.29",
  "tech_stack": ["PHP/7.4", "WordPress 5.8.1"],
  "risk_tags": ["WordPress未授权访问", "..."]
}
```

二、具体对象扫描阶段：针对性漏洞检测

- **目标**
  - **解决问题**：通用扫描工具（如Nikto）误报率高，且无法适配特定技术栈（如WordPress）
  - **创新性**：基于技术栈动态匹配专用工具链，结合实时反馈调整参数，提升检测精度
- **实现方法**
  - **工具链编排**：
    - 根据对象服务要求侦察代理匹配针对性的检测工具

```
if "WordPress" in tech_stack:
    tools = ["wpscan", "nuclei-wordpress"]
elif "PHP" in tech_stack:
    tools = ["sqlmap", "nikto-php-plugins"]
```

    - **并行执行**：使用多线程框架（如Celery）并发运行工具链
  - **动态参数调整**：

        - **场景1：目标响应延迟高**：

```
if avg_response_time > 5s:
    sqlmap_params.update({"--delay": "2", "--threads": "3"})
```

        - **场景2：触发WAF拦截**：

```
sqlmap --random-agent --proxy=http://tor:9050 --chunked
```

    - **自反思与RAG协同优化**
        - **评估指标**：

维度	指标	优化策略
准确性	误报率	启用RAG检索“误报过滤规则”
效率	扫描时间	优化工具并发策略
隐蔽性	IDS告警次数	调用RAG检索“隐蔽扫描参数”

三、深度扫描阶段：针对性深度扫描

- **目标**
    - **解决问题**：传统深度扫描工具（如Nessus）参数固定，无法动态适配目标防御等级，导致效率低下或触发告警
    - **创新性**：基于对象状态（防御等级、服务类型）动态选择扫描工具与参数，平衡覆盖度与隐蔽性
- **实现方法**
    - **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
```

    - **自反思与RAG协同优化**
        - **评估指标**：

维度	量化指标	自反思逻辑示例
覆盖度	漏洞类型数	`if missing_cve_list:` 检索知识库添加检测模板 → 更新nuclei规则库
效率	扫描时间	`if scan_time > 1800s:` 优化工具并发策略 → 减少冗余检测
隐蔽性	IDS告警次数	`if IDS_alerts > 10:` 调用RAG检索“低速随机化参数” → 替换为`nmap -T1`
准确性	误报率	启用RAG误报过滤规则 → 更新工具插件

        - **RAG技术应用**：
            - **知识库构建**：CVE数据库、Exploit-DB、GitHub PoC、历史渗透报告、工具最佳实践参数、隐蔽扫描模板等
            - **检索与生成示例**：

```
# 输入问题：“如何全面扫描Apache服务且避免触发告警”
retrieved_data = rag.query("Apache深度扫描隐蔽参数")
# 返回结果：{"solution": "nmap -T2 --script=http-vuln* --scan-delay 10s"}
```

            - **策略更新**：将检索到的参数动态注入扫描命令。

总结

- **状态驱动的深度扫描**：根据防御等级动态切换工具链，避免资源浪费与告警风险
- **参数自适应优化**：结合环境反馈实时调整扫描强度，平衡效率与隐蔽性
- **知识增强的闭环学习**：通过RAG注入外部最佳实践，持续提升覆盖度与准确性

安全软件信息识别

项目功能

- **输入**：进程名列表/文件名列表/进程模块列表
- **输出**：识别到的安全软件信息

实现过程

- 建立Json格式的数据库，对收集来的数据进行清洗，去从等
- 包含列：模块名，安全软件原名，安全软件原中文名，类别，描述
- 输入支持多种格式：csv，json等；并用json格式输出

Antivirus identification

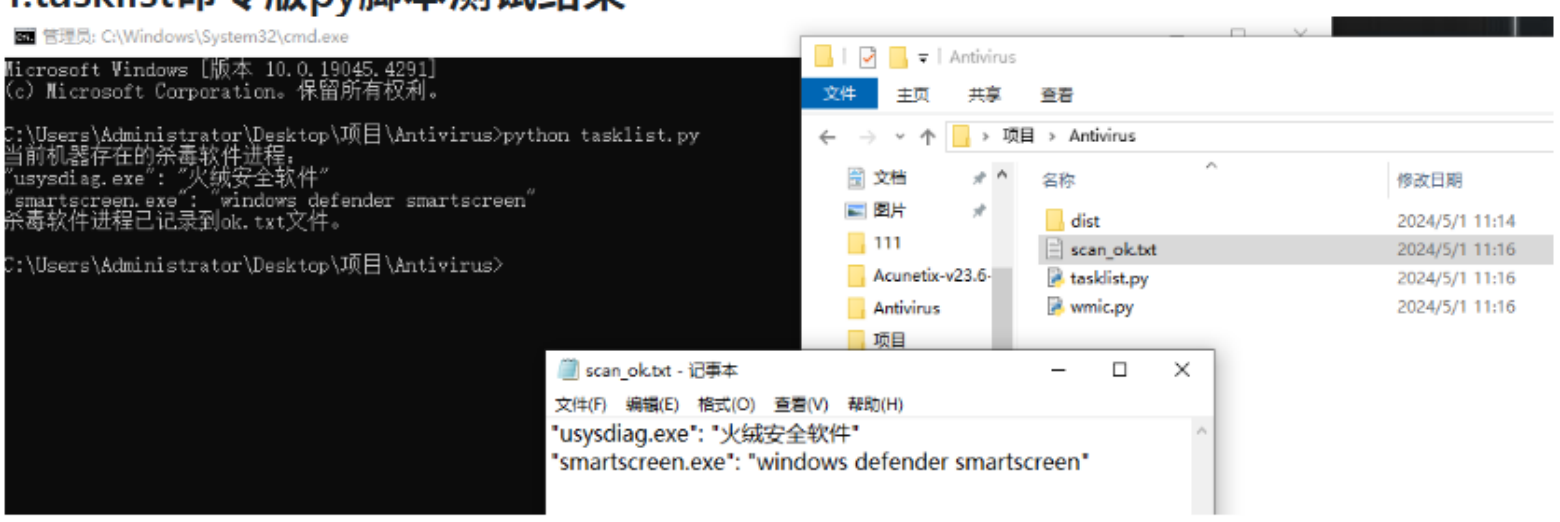
杀毒软件进程识别，脚本内置了相关杀毒软件进程，直接执行即可得知存在什么杀毒软件，不需要自己执行命令。

Anti-virus software process identification, the script built-in anti-virus software process, directly execute to know what anti-virus software exists, do not need to execute their own commands.

因为部分机器不带Python环境，所以编译成了一个exe程序，直接执行exe程序后，会在当前路径下生成一个scan_ok.txt文件，文件内容就是当前机器所存在的杀毒软件

Because some machines do not have a Python environment, it is compiled into an exe program, and after directly executing the exe program, a scan_ok.txt file is generated in the current path, and the file is the anti-virus software existing in the current machine

1.tasklist命令版py脚本测试结果



tasklist命令版exe程序测试结果