

# Project 238 openKylin 操作系统安全分析工具

## 1. 目标描述

“没有网络安全，就没有国家安全”这句话深刻地揭示了网络安全在国家安全战略中的重要地位。在数字化日益深入的今天，网络安全已经成为国家安全的核心组成部分，任何网络空间的漏洞或威胁都可能对国家的政治、经济、军事等领域造成重大影响。当前，国产操作系统在军工、能源、金融、航天、交通等重要领域的应用日益广泛，这不仅标志着我国信息技术产业的飞速发展，也体现了我国在关键信息基础设施自主可控方面取得的显著成就。然而，随着系统的广泛应用，操作系统安全问题也日益凸显，成为行业研究的重点方向。

本次项目的核心目标，正是为了实现“openKylin 操作系统安全分析工具”。这一工具将致力于提升 openKylin 操作系统的安全性，通过深入分析系统的潜在威胁、漏洞和安全风险，为系统管理员和开发者提供全面的安全保障。我们将结合 SCAP 标准/等保安全配置基线标准，进行系统安全评估，对 openKylin 操作系统进行全方位的扫描和检测，确保系统的稳定性和可靠性。

具体而言，该项目将实现以下功能：

- FastScan 版本匹配，快速检测漏洞
- Fofa Fofa 探测
- Iso string Iso 扫描模块
- Nmap 使用 Nmap 模块进行扫描
- OutPutJson string 将漏洞扫描模块输出结果转成 json 文件
- Pdf string 将漏洞扫描模块输出结果转成 pdf 文件
- RA 使用远程检测
- SSHBurst 使用 SSH 爆破
- WKPWD 使用弱口令生成器模块
- all 只扫描 system, kernel 的所有 poc 以及检测 baselin 模块
- baseline string 使用基线检测模块
- ip string 设置 ip, 可设置 ip 段进行验证
- kernel string 使用内核漏洞的验证模块
- repair string 使用基线检测模块
- system string 使用系统漏洞的验证模块
- web string 使用 web 漏洞的验证模块

首先，工具将具备对 openKylin 操作系统的实时监控能力，能够及时发现并报告系统中的安全事件；其次，工具将提供详细的漏洞分析功能，帮助管理员快速定位并修复系统漏洞；最后，工具还将提供安全配置建议和优化方案，帮助管理员提升系统的整体安全水平。

通过本次项目的实施，我们期待能够进一步提升 openKylin 操作系统的安全性，为我国在关键信息基础设施领域的发展提供更加坚实的安全保障。同时，我们也期待通过这一项目的实践，为国产操作系统的安全研究贡献更多的经验和智慧。

## 2. 比赛题目分析和相关资料调研

### 1. 比赛题目概述

本次比赛的核心任务是开发一款针对 openKylin 操作系统的安全分析工具。该工具需要能够对系统进行安全基线检查、漏洞扫描、模块化测试，并提供相应的修复建议。比赛的目标是提高 openKylin 操作系统的安全性，确保系统在面对潜在威胁时能够保持稳定和安全。

本次赛题有两个板块：

- 题目一：实现 openKylin 操作系统安全配置基线检查功能生成可视化报告，并能够根据基线检测结果进行系统安全评估。

其中功能包括以下四点：

- 功能一：实现系统安全配置基线检查
- 功能二：形成安全基线检查报告
- 功能三：系统基线检查安全评估
- 功能四：实现系统总线服务接口自动化安全检查基线

我们需要开发了一个自动化工具，它能够按照预设的安全标准对 openKylin 操作系统的配置进行彻底的检查。

在完成基线检查后，系统会自动生成一份详细的可视化报告。这份报告需要清晰地展示了哪些配置项符合安全基线，哪些需要改进。

基于生成的检查报告，我们的系统需要能够进行深入的安全评估。评估过程涉及对潜在安全风险的分析，并提供了一个综合的安全评分，帮助管理员了解系统的安全状况，并确定需要优先处理的安全问题。

为了进一步加强系统的安全性，我们需要实现对系统总线服务接口的自动化安全检查。这一功能确保了系统的关键通信接口符合安全要求，减少了潜在的安全威胁。

- 题目二：实现 openKylin 操作系统安全漏洞（POC/EXP）检查功能生成可视化报告，能够准确分析出当前操作系统中存在的已公开安全漏洞，并根据漏洞扫描报告进行系统安全评估。

其功能包括以下四点：

- 功能一：实现安全漏洞（POC/EXP）检查功能
- 功能二：形成安全漏洞扫描报告
- 功能三：系统安全漏洞评估
- 功能四：poc 防逆向功能

利用漏洞的 Proof of Concept（POC）和 Exploit（EXP），我们的系统可以模拟攻击行为，以验证漏洞的存在并评估其潜在影响。在完成漏洞检查后，系统将自动生成一份详细的扫描报告。这份报告将包括漏洞的类型、位置、严重性等级以及可能的利用方式，为后续的安全评估和修复提供依据。为了提高系统的安全性，防止恶意用户通过逆向工程获取漏洞的详细信息，我们的系统还具备 poc 防逆向功能。这确保了即使在漏洞被公开的情况下，攻击者也无法轻易地利用这些信息对系统进行攻击。

## 2. 相关资料调研

### 2.1 openKylin 操作系统简介

openKylin 操作系统是一款基于 Linux 的操作系统，它具有高度的可定制性和安全性。了解其架构、版本更新、安全特性以及用户社区是开发安全分析工具的基础。

## 2.2 安全基线检查

安全基线是一组配置设置，它定义了操作系统的`最小安全要求`。安全基线检查是确保操作系统配置符合安全标准的过程。不同的操作系统版本可能有不同的安全配置要求。确认检查的操作系统版本，以确保使用正确的安全基线。调研现有的安全基线标准，例如 CIS（Center for Internet Security）基准，以及如何自动化这些检查，简化基线检查过程，并减少人为错误。

## 2.3 漏洞扫描技术

漏洞扫描是识别系统中安全漏洞的过程。工作原理：漏洞扫描通常基于漏洞数据库，使用扫描和匹配的方式对计算机系统进行脆弱性检测。

## 2.4 模块化测试方法

模块化测试方法是一种软件开发和测试策略，它侧重于将软件系统分解成多个独立的模块或组件，然后分别对这些模块进行测试。这种方法有助于提高测试效率，简化问题定位，并加快开发周期。

## 2.5 修复策略

在发现安全问题后，如何提供有效的修复策略是至关重要的。关键点在于：

**安全配置基线检查：**工具能够实现对 openKylin 操作系统安全配置基线的检查，生成可视化报告，并根据基线检测结果进行系统安全评估。

**生成可视化报告：**检查结果将以可视化报告的形式展现，支持 PDF 和 RTF 格式的报告文档，这有助于用户更直观地理解安全状况。

**修复建议：**工具不仅能够发现安全问题，还能提供对应的修复建议，帮助用户了解如何修补发现的安全漏洞。

**模块化设计：**安全检测功能采用模块化设计，可以通过命令行、命令交互或配置文件进行插拔使用，提高了工具的灵活性和可扩展性。

**跨平台兼容性：**工具考虑了可移植性以及跨平台（CPU 架构）的兼容性，使其能够在不同的系统环境中运行。

**安全漏洞评估：**工具能够根据系统安全漏洞的数量及等级进行系统安全评估，并给出安全评分。

**自动化安全检查：**对于操作系统中的常见系统服务接口，工具能够实现自动化安全检查功能。

**POC 防逆向功能：**为了保护工具中的敏感文件，如 POC/EXP 等不被恶意窃取，工具采用加密、编译、身份认证、内存混淆等方式进行保护。

**开源社区融合：**工具的开发鼓励融合开源社区的优秀项目，但要求这些项目必须是模块化的，以保持工具的模块化特性。

**持续更新：**鉴于安全威胁的不断演变，工具需要定期更新，以包含最新的安全检测技术和修复策略。

## 2.6 相关工具和技术

调研与安全分析相关的工具和技术，例如 genmai 工具的使用：Genmai 内置了丰富的漏洞知识库，包含 15000 多条安全漏洞信息，每条漏洞都有详细的描述和修补建议。Genmai 使用 Go 语言开发，利用其高并发特性，采用 GMP 模式提高并发效率，同时为了避免测试过程中对系统造成影响，Genmai 引入了沙盒模式，对程序和文件进行隔离。

在本地测试场景下，Genmai 设计了反逆向工程的 POC 保护方案，以保护测试用例不被恶意窃取。

配置 genmai 工具的环境时使用到了 Go 语言，需要额外注意版本的匹配问题，版本正确 golang 语句才能正常执行。在配置 python3，导入 python 库时，代理网址出现问题也会导致网址一直拒绝访问，导致无法导入库，这时需要重新设置过后才导入库成功。

我们使用的是 Go 语言，它在系统编程和并发处理方面表现出色。同时将系统设计为模块化，便于维护和扩展。确保开发的工具和报告生成过程本身也是安全的，避免引入新的安全风险。可以考虑提供一个友好的用户界面，让用户能够轻松地执行检查、查看报告和理解评估结果。测试和验证：在开发过程中进行充分的测试，确保功能的正确性和稳定性。

### 3. 系统框架设计

genmai 框架原理是通过先创建 sandbox，将要检测的漏洞 POC/EXP 在 sandbox 上通过 Yaml 解析器以及 Json 解析器将对应文件解析，然后存入缓存中，再通过远程评估和本地评估的形式进行检测并生成分析报告。具体原理如下：

系统将检测以下三种漏洞：系统漏洞（System Vulnerabilities）、内核漏洞（Kernel Vulnerabilities）和 Web 漏洞（Web Vulnerabilities）。具体检测流程如下：

#### 1. 漏洞检测与解析

- 系统漏洞和 Web 漏洞的数据将通过 JSON 解析器（Json Parser）进行解析。
- 内核漏洞的数据将通过 YAML 解析器（Yaml Parser）进行解析。

#### 2. 数据缓存

- 解析后的漏洞数据将被送入 缓存（Cache）模块进行缓存。

#### 3. 本地处理

- 缓存的数据会被发送到 本地处理（Local）模块进行处理。

#### 4. 数据处理与转发

- 本地处理（Local）模块处理完数据后，将数据分别发送到 协程池（Coprogram Pool）和请求池（Request Pool）。
- 协程池（Coprogram Pool）负责处理本地处理后的数据。
- 请求池（Request Pool）负责处理并转发需要进行远程评估的数据。

#### 5. 远程评估

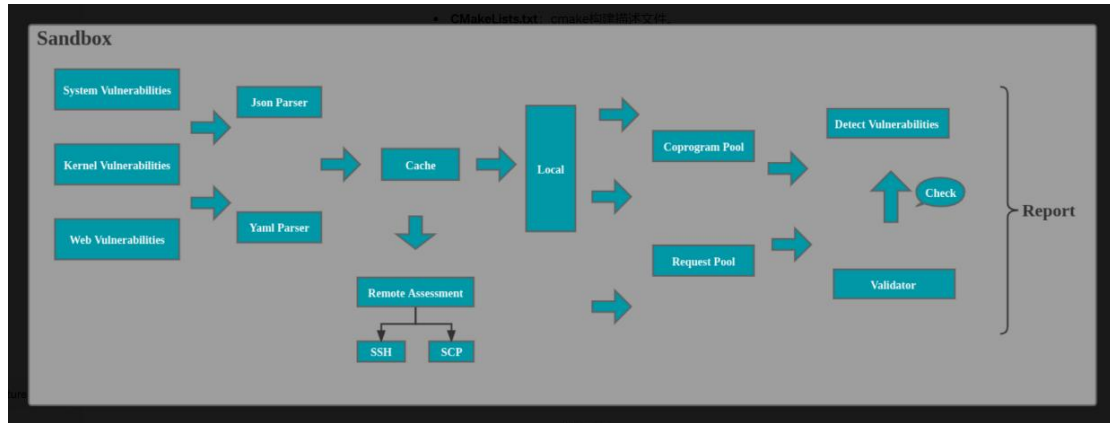
- 请求池（Request Pool）转发的数据将通过 SSH 和 SCP 进行 远程评估（Remote Assessment）。

#### 6. 漏洞检测与验证

- 漏洞检测（Detect Vulnerabilities）通过检查和验证来检测漏洞：
- 先通过 检查（Check）模块对检测到的漏洞进行检查。
- 然后通过 验证器（Validator）验证漏洞的准确性和有效性。

#### 7. 报告生成

- 最终生成 报告（Report），即完整的漏洞检测报告。



该项目结构包含以下各个目录和文件：

Genmai:

frontend: 前端 UI 目录，包含 HTML、CSS、JS 等前端语言文件。

app: 应用层目录，使用 Go 语言编写。

core: 核心层目录，使用 Go 语言编写。

infra: 基础设施层目录，使用 Go 语言编写。

tools: 工具层目录，使用 Go 语言编写。

data: 数据层目录，包含数据库文件、Python 脚本、配置文件和规则文件。

debian: debian 包构建目录，包含构建 debian 包所需的流程、规则和控制文件。

genmai.spec: rpm 包构建文件，包含 rpm 包构建所需的包描述文件。

CMakeLists.txt: cmake 构建描述文件。

- \* 类型、类、外部函数采用大驼峰命名法
- \* 变量、内部函数采用小驼峰命名法

#### 4. 开发计划

我们将采用所提供的 Genmai 框架，去对项目进行开发：

1. 实现系统安全配置基线检查，检查项覆盖操作系统通用安全需求(参考 SCAP 标准/等保安全配置基线标准)。
2. 形成安全基线检查报告，生成可视化报告，支持 pdf 和 rtf 格式报告文档。
3. 实现安全漏洞（POC/EXP）检查功能，检查功能模块化，基于安全漏洞 poc/exp 检测。
4. 形成安全漏洞扫描报告，生成可视化报告，支持 pdf 和 rtf 格式报告文档。

#### 5. 比赛过程中的重要进展

1. 创建 Kylin 操作系统的运行环境。
2. 配置 Genmai 运行环境，并且成功运行。
3. 实现系统安全配置基线检查，检查项覆盖操作系统通用安全需求。
4. 实现安全漏洞检查功能，检查功能模块化。
5. 能够生成 Pdf 或 rtf 格式文本。

#### 6. 系统测试情况

## 1. 系统安全配置基线检查

测试基线检查：在终端输入 `go run main.go -baseline=all -passwd=*****`（密码是系统的 root 密码）开始进行基线检查

截图：

```
Baseline info: non-exists
../data/SandboxViPyEnv/amd64/myenv/bin/python3: error while
le: No such file or directory

Baseline info: non-exists
../data/SandboxViPyEnv/amd64/myenv/bin/python3: error while
le: No such file or directory

Baseline info: non-exists
=====Baseline Scan Results=====

time 2024-05-30 09:44:58
BaselineExecPoc: 107
BaselineRepairedNums: 107
BaselineNotFixedNums: 0
BaselineNotExecPocNums: 0
Baseline info: No vulnerability
[*] Restoring sshd configuration file...
[*] Restarting sshd service...
[*] Restoring root password...
root@ljc-vmwarevirtualplatform:/home/ljc/genmai/src#
```

分析报告：

图中扫描了 107 项基线项、修复了 107 项基线项、未修复的基线项为 0、未扫描的基线项为 0。

## 2. 安全漏洞（POC/EXP）检查功能

模块化测试：

### 2.1 Kernel 模块

在终端输入 `go run main.go -kernel=all` 开始进行 poc/exp 检测内核模块是否存在漏洞

截图：

```
*] info:io: >.:
*] info:io: >?: root
*] info:a ou:
*] info:exploreWithPath(): >?: CVE-2022-2639 Check poc failed!
kernel info CVE-2022-2639 Check poc failed!
=====Kernel Scan Results=====

kernelExecPoc: 6
kernelRepairedNums: 5
kernelNotFixedNums: 0
kernelNotExecPocNums: 1
kernel info: No vulnerability
root@ljc-vmwarevirtualplatform:/home/ljc/genmai/src#
```

分析报告：

图中扫描了 6 项内核项、修复了 5 项内核项、未修复的内核项为 0、未扫描的内核项为 0。

### 2.2 System 模块

在终端输入 `go run main.go -system=all` 开始系统检测

截图：

```
[*] System info: CVE-2022-1771 Check poc failed!
[*] info:io: ??: 0
[*] info:a ou:
[-] System info: CVE-2022-0572 Check poc failed!
[*] info:io: ??: 0
[*] info:a ou:
warning:ID: CVE-2022-0351 takes too long! (Expiredtime = 10 seconds)
[-] System info: CVE-2022-0351 Check poc failed!

=====漏洞POC验证检测结果=====

完成检测时间: 2024-05-30 10:03:11
已执行POC数量: 29
已修复漏洞数量: 27
未修复漏洞数量: 2
未执行POC数量: 0
root@ljc-vmwarevirtualplatform:/home/ljc/genmai/src#
```

分析报告：

图中执行了 29 项 POC、修复了 27 个漏洞、  
未修复的漏洞数为 0、未执行的 POC 为 0。

## 7. 遇到的主要问题和解决方法

### 1. OpenKylin 操作系统的配置

因为本次我们的比赛赛题是基于 Openkylin 操作系统下进行操作，所以我们需要先配置好 OpenKylin 操作系统，在这个过程中我们也是遇到了许多自己无法解决的问题，普遍的解决方法还是在网上查找资料进行解决。

### 2. genmai 工具的环境搭建

配置 genmai 工具的环境时，比如在配置 go 语言时遇到了明明已经配置了 golang，但是就是运行不了 go，后面发现是版本的问题，解决方法就是重新下载一个符合版本的 golang。在配置 python3，导入 python 库时，就出现了网址一直在拒绝访问，导致无法导入库，后面通过上网查询资料发现是代理网址的问题，重新设置过后才导入库成功。

3. Pdf 模块生成 pdf 文件，可以单独执行生成 pdf 文件，但无法对模块进行直接调用生成新 pdf 文件。

我们选择先写一个 pdf 模板，当生成漏洞报告时，把生成的参数传入并修改 pdf 文档，生成 pdf 报告。

4. 在调用生成 rtf 格式文件时，无法下载库函数。

配置 GitHub 用户名密码，有权限访问 <https://github.com/knowlet/rtf> 登录 GitHub 后解决。

5. 模块化调用的不规范，在运行时无法进行命令行控制。

我们选择在命令行携带参数，分别调用不同模块，使得功能模块化。

## 8. 分工和协作

小组一共有三人，分别是组长朱泓印，组员：李进超、麦昊。

以下简述组内各自完成的任务：

朱泓印：环境的搭建，功能模块化，生成漏洞报告 pdf 功能实现，编写赛道文档

李进超：环境的搭建，模块化功能实现，编写赛道文档

麦昊：环境的搭建，编写赛道文档

## 9. 提交仓库目录和文件描述

src 为整体的项目功能展示，pdf 为项目文档。

## 10. 比赛收获

参加了本次比赛我们收获了许多，深入理解了操作系统的核心概念，学会了应用高级操作系统技术，通过在实际项目中运用理论知识，我的问题解决能力得到了显著提升。在实际项目中应用理论知识，解决实际问题的能力提升，通过编写代码和调试程序，提高了编程技能和问题解决能力。在团队项目中与队友密切合作，分工协作，提升了团队合作能力，通过讨论和交流，增强沟通能力，学会如何在团队中表达自己的想法和建议。这些收获不仅有助于提升技术水平，还能为未来的职业发展打下坚实的基础。

