readme

Rust 项目依赖漏洞扫描工具

这是一个用于扫描 Rust 项目依赖中潜在安全漏洞的命令行工具。它支持扫描本地项目或从 GitHub 下载的项目 ZIP 文件。

功能特点

- 支持扫描本地项目的 Cargo.lock 文件
- 自动从 ZIP 压缩包中提取和分析项目文件
- 支持直接使用 GitHub 项目下载的 ZIP 文件
- 使用官方 RustSec Advisory DB 进行漏洞检测
- 如果项目中没有 Cargo.lock 文件，会尝试自动生成
- 生成详细的 JSON 格式漏洞报告
- 扫描完成后自动清理临时文件
- 支持生产 json 格式的 sbom 文件

使用方法

```
cargo run -- <path-to-zip-file>
```

例如：

```
# 扫描本地项目的 ZIP 文件
cargo run -- ./test/myproject.zip

# 扫描从 GitHub 下载的项目
cargo run -- ./downloads/some-project-main.zip

# 在demo中演示
cargo run -- .test/demo/project.zip
```

输出说明

工具会生成一个 JSON 格式的漏洞报告，包含以下信息：

- 扫描的包总数
- 发现的漏洞数量
- 按严重程度分类的漏洞统计
- 每个漏洞的详细信息（包名、版本、漏洞描述等）

报告将保存在 `./output/vuln_report.json` 文件中。

工具会生成一个 JSON 格式的sbom 文件，保存在 `./output/sbom.json` 文件中

项目结构

extract_zip.rs

用于从压缩包中提取 Cargo.lock、Cargo.toml 等项目文件。支持处理常规项目结构和 GitHub 下载的 ZIP 文件格式。

get_lockfile.rs

处理 Cargo.lock 文件的加载和解析。如果项目中没有 Cargo.lock 文件，会尝试自动生成。主要功能：

- 从压缩包中提取并解析 Cargo.lock
- 识别项目根目录结构
- 自动处理缺失的 lock 文件

scanner.rs

核心扫描逻辑，使用 RustSec Advisory DB 检查依赖中的已知漏洞。

main.rs

程序入口，处理命令行参数并协调整个扫描流程。

依赖说明

- cargo-lock：解析 Cargo.lock 文件
- rustsec：漏洞数据库和检查
- anyhow：错误处理
- serde：JSON 序列化
- walkdir：文件系统遍历
- zip：处理 ZIP 文件

注意事项

1. 确保 `./data/advisory-db` 目录存在且包含最新的 RustSec Advisory DB
2. 临时文件会被存放在 `./tmp` 目录，扫描完成后自动清理
3. 漏洞报告默认输出到 `./output` 目录
4. `./demo`：演示用 ZIP（`project.zip`）与示例项目目录（`demo_hello`）。
5. 确保要检测的项目，已经包含 lock 文件，如果没有请先 cargo build 或 cargo generate-lockfile