**构建、使用文档**

**一、构建方法**

**代码仓说明**

本项目共有两个代码仓：

patch\_dataset：包含真实世界中收集的100个 C/CPP 与50个 Golang CVE补丁backport迁移失败案例。为每个项目的每个CVE单独准备数据，其中包含失败信息，编译脚本，testsuite脚本(部分)，PoC脚本(部分)。

patch-backporting：项目源码，其中包含与大模型交互代理及工具。

**源码目录结构**

|  |
| --- |
| Plain Text . ├── README.md ├── logs ├── pdm.lock ├── pyproject.toml ├── requirements.txt ├── src │ ├── agent │ │ ├── \_\_init\_\_.py │ │ ├── invoke\_llm.py │ │ └── prompt.py │ ├── backporting.py │ ├── check │ │ ├── \_\_init\_\_.py │ │ └── usage.py │ ├── example.yml │ └── tools │ ├── \_\_init\_\_.py │ ├── logger.py │ ├── project.py │ └── utils.py └── test  ├── test\_hunk.py  └── test\_patch.py |

**软件依赖**

Python >= 3.10

Ctags 5.9.0 sudo apt install universal-ctags

PDM 2.18.2 https://github.com/pdm-project/pdm

**依赖配置方法**

1. 安装PDM

|  |
| --- |
| Bash $ curl -sSL https://pdm-project.org/install-pdm.py | python3 - |

1. 配置Python及其依赖

|  |
| --- |
| Bash # 在patch-backporting路径下 $ pdm install $ source .venv/bin/activate |

**二、使用方法**

**数据配置**

在开始测试前需要准备两部分数据，其一是config.yml包含backport的相关项目信息，其二是目标backport补丁相关的编译、测试脚本。

1. 配置信息

|  |
| --- |
| YAML # example config yaml project: libtiff project\_url: https://github.com/libsdl-org/libtiff  new\_patch: 881a070194783561fd209b7c789a4e75566f7f37 # patch commit id in new version, Version A(Fixed)  new\_patch\_parent: 6bb0f1171adfcccde2cd7931e74317cccb7db845 # patch parent commit, Version A  target\_release: 13f294c3d7837d630b3e9b08089752bc07b730e6 # commid id which need to be fixed, Version B  sanitizer: LeakSanitizer # sanitizer type for poc, could be empty error\_message: "ERROR: LeakSanitizer" # poc trigger message for poc, could be empty tag: CVE-2023-3576 openai\_key: # Your openai key project\_dir: dataset/libsdl-org/libtiff # path to your project source code patch\_dataset\_dir: ~/backports/patch\_dataset/libtiff/CVE-2023-3576/  # path to your patchset, include build.sh, test.sh ....  # Version A Version A(Fixed)  # ┌───┐ ┌───┐ ┌───┐  # │ ├───────────►│ ├────────────►│ │  # └─┬─┘ └───┘ └───┘  # │  # │  # │  # │ Version B  # │ ┌───┐  # └─────────────►│ ├────────────► ??  # └───┘ |

1. 相关脚本

为了保证验证链的正常运行，需要提供编译、测试、PoC脚本在patch\_dataset\_dir所指定的目录中。如下目录所示，编译脚本命名为build.sh，测试脚本为test.sh，poc.sh为当前CVE的PoC触发脚本。并将脚本中所需的文件一起放入该目录中，工具在大模型生成补丁后会自动调用验证。

|  |
| --- |
| Go CVE-2023-3576 ├── build.sh ├── config.yml ├── poc ├── poc.sh └── test.sh |

当然，以上脚本缺失时本工具也可正常运行，但编译脚本可以为backport提供更多可能的信息。

**测试**

|  |
| --- |
| Bash python backporting.py --config example.yml --debug # python backporting.py --config YOUR\_CONFIG [--debug] # python backporting.py -c YOUR\_CONFIG [-d] |

测试时按如上命令执行src下的backporting脚本进行补丁迁移，在example.yml处替换为配置好的目标yaml文件。同时，debug mode会提供更多与大模型交互的信息，以便查看迁移过程。在非debug模式下的测试输出如下：

