补丁定位器

功能简介

Phase 1: Collect disclosed vulnerabilities and multi-branch

> 这个功能是为了从安全网站中爬取漏洞信息，包括CVE ID、漏洞描述、漏洞类型、漏洞发布时间、漏洞更新时间、漏洞影响的软件、漏洞的补丁文件等信息；并且从GitHub上爬取目标存储库的分支信息。

1. 爬取漏洞信息
    - ~/data/download_and_parse_NVD.download_NVD_data_feed(start_year,end_year) 这个文件是通过爬取网站链接'https://nvd.nist.gov/feeds/json/cve/1.1/nvdcve-1.1-%s.json.gz'来获取漏洞信息的。其中%s是年份，start_year和end_year是爬取漏洞信息的起始年份和结束年份。
        - requiere：start_year、end_year
            - start_year：选择开始年份
            - end_year：选择结束年份
        - output：漏洞信息
            - CVE ID
            - problemtype：漏洞的类型
            - patch：漏洞的补丁文件及其链接
            - Description：漏洞的描述
            - configurations:Affected Software etc.
            - impact: 漏洞的影响程度
            - Published Date：漏洞的发布时间
            - Last Modified Date：漏洞的更新时间
            - example: 这是一个Windows Print Spooler服务中的漏洞，它允许攻击者在系统上执行任意代码。
                - CVE ID: CVE-2021-34527
                - problemtype: CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer
                - patch: Microsoft has released a security update to address this vulnerability.
                - Description: A remote code execution vulnerability exists when the Windows Print Spooler service improperly performs privileged file operations. An attacker who successfully exploited this vulnerability could run arbitrary code with SYSTEM privileges. An attacker could then install programs; view, change, or delete data; or create new accounts with full user rights.
                - configurations:Affected Software: Windows Server 2019 etc.

        - impact: Critical
        - Published Date: 2021-06-08
        - Last Modified Date: 2021-06-09
    - run example: 'python download_and_parse_NVD.py'

Phase 2: Collect natural patches

> 这个功能是为了从NVD安全网站中在线爬取补丁文件信息，包括补丁文件的commit、补丁文件的commit时间等信息。

1. 多进程爬取补丁信息:~/tracer-master/vulnerability_analysis/patch_localization/tool/tool_main.py
    - run_process() 多进程爬取指定年份CVE的补丁信息
    - crawl_cve(START_YEAR,END_YEAR)
        - requiere：mode、START_YEAR、END_YEAR
            - START_YEAR：选择开始年份
            - END_YEAR：选择结束年份

Real Patch Collection and Network Construction

> 这个功能是为了从github、NVD、RED HAT等漏洞信息存储库中在线爬取等价的补丁commit信息，并构建CVE-PATCH映射图。

- ~/tracer-master/vulnerability_analysis/patch_localization/tool/tool_main.py
    - start(CVEID,year)
        - requiere：CVEID、year
            - CVEID：CVE ID
            - year：选择年份
        - steps： step1. 初始化CVE对象 step2: 读取NVD、Redhat、Debian等安全网站信息,并解析出补丁的参考链接refs, 并放入graph中 step3: 识别目标节点类型(target_types_of_nodes)，放入graph中 step4: 递归分析补丁url节点信息 step5: 提取graph中 CVEID、Issuekey、github_repo信息 step6: 在Github中搜索补丁commit,并加入图中
        - output：以CVE为根节点的CVE-PATCH映射图
        - run example: 'python tool_main.py'

Locating Security Patches on Multibranch Software

> 这个功能是为了能够在目标存储库中定位NVD中的补丁文件。

1. 多进程运行补丁定位程序Overall_patch_locator.py
    - requiere：mode、START_YEAR、END_YEAR
        - mode：选择是定位NVD中的补丁文件还是定位目标存储库中的补丁文件

        - START_YEAR：选择开始年份
        - END_YEAR：选择结束年份
2. 补丁定位程序 Patch_locator.patchlocator(repo,branch,patchesinfo_path,year,cve_ID,output_dir)
    - requiere：targetrepo、targetbranch、patchesinfo
        - targetrepo：目标代码仓库的路径
        - targetbranch：目标分支的名称
        - patchesinfo_path：包含需要定位的补丁文件信息的文件路径
    - optional：year、cve_ID、output_dir
        - year：选择年份
        - cve_ID：选择CVE ID
        - output_dir：补丁定位结果的输出目录
    - output：定位到的补丁文件的信息,如果根据严格匹配规则定位到了补丁commit，则保存补丁commit的相关信息;./output/[year]/[cve_ID]]/[owner]%[repo]/[branch] (./output/2011/CVE-2011-0007/troglobit%pimd/dev)
        - cve_ID
        - commit
        - commit Date
        - example:
            - CVE-2011-0007 b0bde380abad (2011, 1, 8)
            - CVE-2011-0007 35606839f549 (2011, 1, 9)
            - CVE-2011-0007 b0bde380abad (2011, 1, 8)
    - run example: 'python Patch_locator.py /home/user/repo master /home user/patches.txt'

Phase 3: Synthetic Semantic Equivalent patches

> 这个功能是为了在目标存储库中定位的补丁commit的基础上，通过3种方法生成语义等效的补丁commit。

- Patch_evolution.Patchevolution_tracker(repo,branch,patchesinfo_path,cve_id,output_dir,year,save_source_path)
    - requiere：targetrepo、targetbranch、patchesinfo
        - repo：目标代码仓库的路径
        - branch：目标分支的名称
        - patchesinfo_path：包含需要定位的补丁文件信息的文件路径
    - optional：year、cve_id、output_dir
        - year：选择年份
        - cve_ID：选择CVE ID
        - output_dir：补丁定位结果的输出目录
        - save_source_path：保存补丁commit的路径
    - output：
        - 自然演化等价补丁
        - AST控制流等价补丁

Equivalent Natural Evolution of Real security patches

> 这个功能是通过自然演化的方式生成语义等效的补丁commit。

1. 寻找aftercommit: 通过定位补丁commit找到其之后的一定数量的aftercommits;
    - get_afterpatchcommits(repopath,branch,filename,yi_maincommit)
        - requiere：repopath、branch、filename、yi_maincommit
            - repopath：目标代码仓库的路径
            - branch：目标分支的名称
            - filename：需要定位的补丁文件信息的文件名称
            - yi_maincommit：定位到的补丁commit
        - output：aftercommits
    - 通过补丁commit Date来选择aftercommits
        - requiere：chosencommits、yi_maincommit、patch_time
            - chosencommits：aftercommits
            - yi_maincommit：补丁commit
            - patch_time：补丁commit Date

    output: aftercommits

2. 寻找beforecommit
    - helper_zz.get_bereocommit2(repopath, yi_maincommit, branch, filename)
        - requiere：repopath、yi_maincommit、branch、filename
            - repopath：目标代码仓库的路径
            - yi_maincommit：定位到的补丁commit
            - branch：目标分支的名称
            - filename：需要定位的补丁文件信息的文件名称
        - output：beforecommits
3. 产生Natural Evolution等价补丁
    - save_after_func(repo, branch, yi_commit, afterpatchcommit,beforepatchcommit,yi_cve,cve_commit_element_content,save_path)
        - requiere：repo、branch、yi_commit、afterpatchcommit、beforepatchcommit、yi_cve、cve_commit_element_content、save_path
            - repo：目标代码仓库的路径
            - branch：目标分支的名称
            - yi_commit：定位到的补丁commit
            - afterpatchcommit：aftercommits
            - beforepatchcommit：beforecommits
            - yi_cve：CVE ID
            - cve_commit_element_content：补丁commit的内容
            - save_path：保存补丁commit的路径
        - output：aftercommit的补丁commit函数

- save_before_func(repo, branch, yi_commit, beforepatchcommit, yi_cve, cve_commit_element_content, save_path)
  - requiere：repo、branch、yi_commit、beforepatchcommit、yi_cve、cve_commit_element_content、save_path
    - repo：目标代码仓库的路径
    - branch：目标分支的名称
    - yi_commit：定位到的补丁commit
    - beforepatchcommit：beforecommits
    - yi_cve：CVE ID
    - cve_commit_element_content：补丁commit的内容
    - save_path：保存补丁commit的路径
  - output：beforecommit的补丁commit函数
- get_real_func_diff(repo, branch, yi_commit, diff_file_name, function_dic, save_path)
  - requiere：repo、branch、yi_commit、diff_file_name、function_dic、save_path
    - repo：目标代码仓库的路径
    - branch：目标分支的名称
    - yi_commit：定位到的补丁commit
    - diff_file_name：补丁commit的diff文件名称
    - function_dic：补丁commit的函数字典
    - save_path：保存补丁commit的路径
  - output：自然演化补丁commit的函数diff文件

Equivalent AST Control Flow of Real security patches

> 这个功能是通过AST控制流的方式生成语义等效的补丁commit。

- get_equal_funcdiff(repo, branch, yi_commit, diff_file_name, function_dic, save_path)
  - requiere：repo、branch、yi_commit、diff_file_name、function_dic、save_path
    - repo：目标代码仓库的路径
    - branch：目标分支的名称
    - yi_commit：定位到的补丁commit
    - diff_file_name：补丁commit的diff文件名称
    - function_dic：补丁commit的函数字典
    - save_path：保存补丁commit的路径
  - output：6种AST控制流等价补丁commit的函数diff文件
- get_patch_var(equal_path,before_func_first, after_func_first,index)
  - requiere：equal_path、before_func_first、after_func_first、index
    - equal_path：AST控制流等价补丁commit的函数diff文件
    - before_func_first：beforecommit的第一个函数
    - after_func_first：aftercommit的第一个函数
    - index：等价补丁commit的类型索引
  - output：一种AST控制流等价补丁commit的函数diff文件，类型由index决定

Equivalent Semantic Fixes of Real security patches

> 这个功能是通过编译-反编译的方法生成与真实补丁修复能力一致的补丁

- ~/tracer-master/Patchlocator-master/Patch_compile.py
  - Patchevolution_tracker(repo, branch, patches_info_path, cve_id=None, output_dir=None, year=None)
    - requiere：repo、branch、patches_info_path、cve_id、output_dir、year
      - repo：目标代码仓库的路径
      - branch：目标分支的名称
      - patches_info_path：补丁信息文件的路径
      - cve_id：CVE ID
      - output_dir：保存补丁commit的路径
      - year：CVE年份
    - output：补丁commit的编译-反编译信息
    - steps
      - Step1 寻找natural patch 的aftercommit
      - Step2 对于每一个aftercommit，执行一次reset，然后执行一次make编译，然后执行一次反编译
        - Step2.1 执行reset
        - Step2.2 执行make编译
        - Step2.3 执行反编译
        - Step2.4 保存补丁commit的编译后的.c 和.o文件
      - Step4 对于每一个aftercommit,执行一次diff，然后执行一次patch
    - run command
      - python3 ~/tracer-master/Patchlocator-master/Patch_compile.py [repo] [branch] [patch_info_path]

> Tips: 1. 这个版本的编译与反编译脚本需要手动输入补丁的Repo、branch、patchinfo信息。 2. 在编译与反编译过程中，脚本中使用了git reset commit命令，会导致git log中信息的丢失。如果执行了git reset commit命令，在该commit之后的所有git log信息将会丢失。需要谨慎使用。

许可证