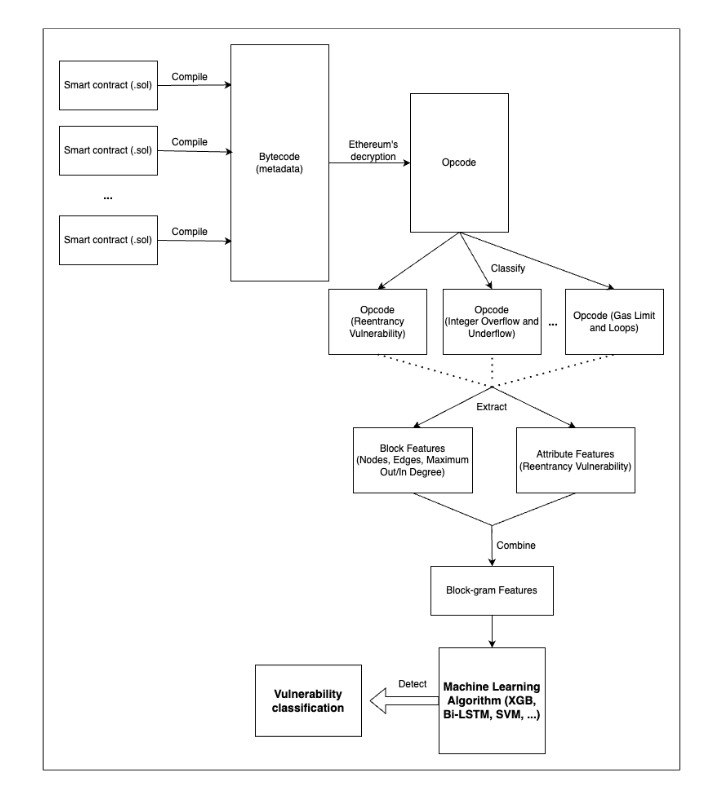
Đề tài: PHÁT TRIỂN CÔNG CỤ HỖ TRỢ PHÁT HIỆN LỖ HỔNG BẢO MẬT CHO ỨNG DỤNG WEB3 DÀNH CHO CÁC NHÀ PHÁT TRIỂN

Developing a Tool to Support Security Vulnerability Detection for Web3 Applications for Developers



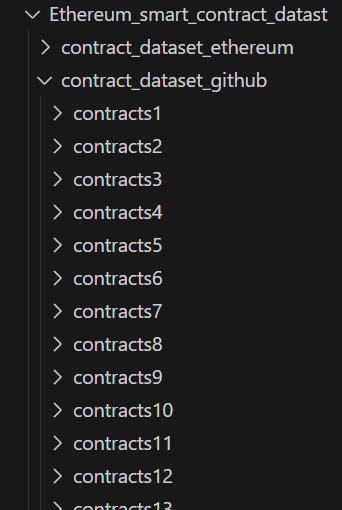
datasets thì mình chỉ lấy của 3 loại theo sơ đồ, gom trêncác nguồn Smartbug, SolidiFI, Messsi-Q.

- <https://github.com/Messi-Q/Smart-Contract-Dataset?tab=readme-ov-file>

- Chay tren VS Code:

+ Tải npm, nodejs: sudo apt install nodejs npm

+ Tải thư viện và chỉnh sửa: (sẽ có nhiều file lỗi, vì k tìm được thư viện, với dataset lớn thì mấy cái đó mình giả bộ quên nó đi)

* Đa phần dataset chứa nhưng file sol sử dụng thư viện khá cũ (để có lỗ hổng) (<5.0.0 version solidity) -> thư viện sử dụng là openzeppelin\_contracts version dưới 2.3.0 thì mới có thể chạy được  
  -> Nhưng hiện tại các bản thư viện đó đã cũ, nên phải tự tải lại lên github: <https://github.com/OpenZeppelin/openzeppelin-contracts/releases>
* Chuyen sol qua bytecode (hien tai là đang ở dataset này): 
* Code chuyen bytecode :

import os

import subprocess

import re

from packaging import version

def run\_solc\_and\_save(source\_directory, destination\_directory):

    # Get all .sol files in source\_directory

    filenames = [f for f in os.listdir(source\_directory) if f.endswith('.sol')]

    for filename in filenames:

        file\_path = os.path.join(source\_directory, filename)

        # Determine the appropriate Solidity version

        mid\_version = "0.8.26"  # Default version if no pragma found

        with open(file\_path, 'r', encoding='latin1') as f:

            lines = f.readlines()

            pragma\_lines = [line for line in lines if line.strip().startswith("pragma solidity")]

            versions = []

            try:

                for pragma\_line in pragma\_lines:

                    versions.extend(re.findall(r'\d+\.\d+\.\d+', pragma\_line))

                if versions:

                    mid\_version = max(versions, key=lambda v: version.parse(v))

            except Exception as e:

                print(f"Error parsing Solidity version for {filename}: {str(e)}. Using default version {mid\_version}.")

        # Check if the required Solidity version is installed

        try:

            subprocess.run(["solc-select", "install", mid\_version], check=True)

        except subprocess.CalledProcessError:

            print(f"Failed to install Solidity version {mid\_version}. Skipping {filename}.")

            continue

        # Use the appropriate Solidity version

        try:

            subprocess.run(["solc-select", "use", mid\_version], check=True)

        except subprocess.CalledProcessError:

            print(f"Failed to switch to Solidity version {mid\_version}. Skipping {filename}.")

            continue

        # Create the path for the bin file

        bin\_file\_path = os.path.join(destination\_directory, os.path.splitext(filename)[0])

        if not os.path.exists(bin\_file\_path):

            os.makedirs(bin\_file\_path)

        # Skip if the bin file already exists

        if os.path.exists(os.path.join(bin\_file\_path, f"{os.path.splitext(filename)[0]}.bin")):

            print(f"Bin file for {filename} already exists. Skipping...")

            continue

        # Run solc and save bytecode

        try:

            subprocess.run(

                ["solc", "--bin", file\_path, "-o", bin\_file\_path],

                check=True,

            )

            print(f"Saved bytecode for {filename} to {bin\_file\_path}")

        except subprocess.CalledProcessError as e:

            print(f"Solc could not run on {filename}: {e}. Skipping...")

# Đường dẫn nguồn và đích

SOURCE\_DIR = r"/root/devCute/Ethereum\_smart\_contract\_datast/contract\_dataset\_github/contracts16/"

DESTINATION\_DIR = r".\bytecode\_output\contracts16"

# Chạy chương trình

run\_solc\_and\_save(SOURCE\_DIR, DESTINATION\_DIR)

Bai bao:

a. P. K. Ghaleb, "How effective are smart contract analysis tools? evaluating smart contract static analysis tools using bug injection," in Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis, 2020, pp. 415--427.

P. a. L. Z. a. H. Q. a. Z. R. a. W. X. Qian, "Towards automated reentrancy detection for smart contracts based on sequential models," IEEE Access, vol. 8, pp. 19685--19695, 2020.

S. S. a. J. S. a. S. D. a. K. M. a. L. H.-N. Kushwaha, "Systematic review of security vulnerabilities in ethereum blockchain smart contract," IEEE Access, vol. 10, pp. 6605--6621, 2022.

S. a. R. A. a. P. C. Akca, "SolAnalyser: A framework for analysing and testing smart contracts," in 2019 26th Asia-Pacific software engineering conference (APSEC), IEEE, 2019, pp. 482--489.

N. a. K. M. a. J. A. a. B. L. a. S. B. a. S. Y. Grech, "Madmax: Surviving out-of-gas conditions in ethereum smart contracts," Proceedings of the ACM on Programming Languages, vol. 2, pp. 1--27, 2018.

J. a. G. G. a. G. A. Feist, "Slither: a static analysis framework for smart contracts," in 2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB), IEEE, 2019, pp. 8--15.

S. a. V. E. a. I. I. a. T. R. a. M. E. a. A. Y. Tikhomirov, "Smartcheck: Static analysis of ethereum smart contracts," in Proceedings of the 1st international workshop on emerging trends in software engineering for blockchain, 2018, pp. 9--16.

M. a. M. F. a. H. E. a. G. A. a. G. G. a. F. J. a. B. T. a. D. A. Mossberg, "Manticore: A user-friendly symbolic execution framework for binaries and smart contracts," in 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE, 2019, pp. 1186--1189.

P. a. D. A. a. D.-C. D. a. G. A. a. B. F. a. V. M. Tsankov, "Securify: Practical security analysis of smart contracts," in Proceedings of the 2018 ACM SIGSAC conference on computer and communications security, 2018, pp. 67--82.

I. a. M. X. a. J. B. a. C. W. K. Ashraf, "GasFuzzer: Fuzzing ethereum smart contract binaries to expose gas-oriented exception security vulnerabilities," IEEE Access, vol. 8, pp. 99552--99564, 2020.

T. D. a. P. L. H. a. S. J. a. L. Y. a. M. Q. T. Nguyen, "sfuzz: An efficient adaptive fuzzer for solidity smart contracts," in Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering, 2020, pp. 778--788.