

# 기술문서 최종발표

2023 KEEPER Technical Research Paper

# 목차

- 공부한 내용
- 기술문서 결과
- 다음 계획

- 블록체인 취약점들

- Improper Input Validation
- Incorrect Calculation
- Oracle/Price Manipulation
- Weak Access Control
- Replay Attack/Signature Malleability

- 취약점 실습

- Ethernaut



- 도구

- Foundry

- Rounding Error
- Reentrancy
- Front-Running
- Uninitialized Proxy
- Governance Attacks

- Damn Vulnerable Defi

## Ethernaut, DVD

-  Solidity 코드 이해
-  취약한 코드 패턴 학습
- ✨ 다른 체인에서도 적용 가능

```
pragma solidity ^0.8.0;

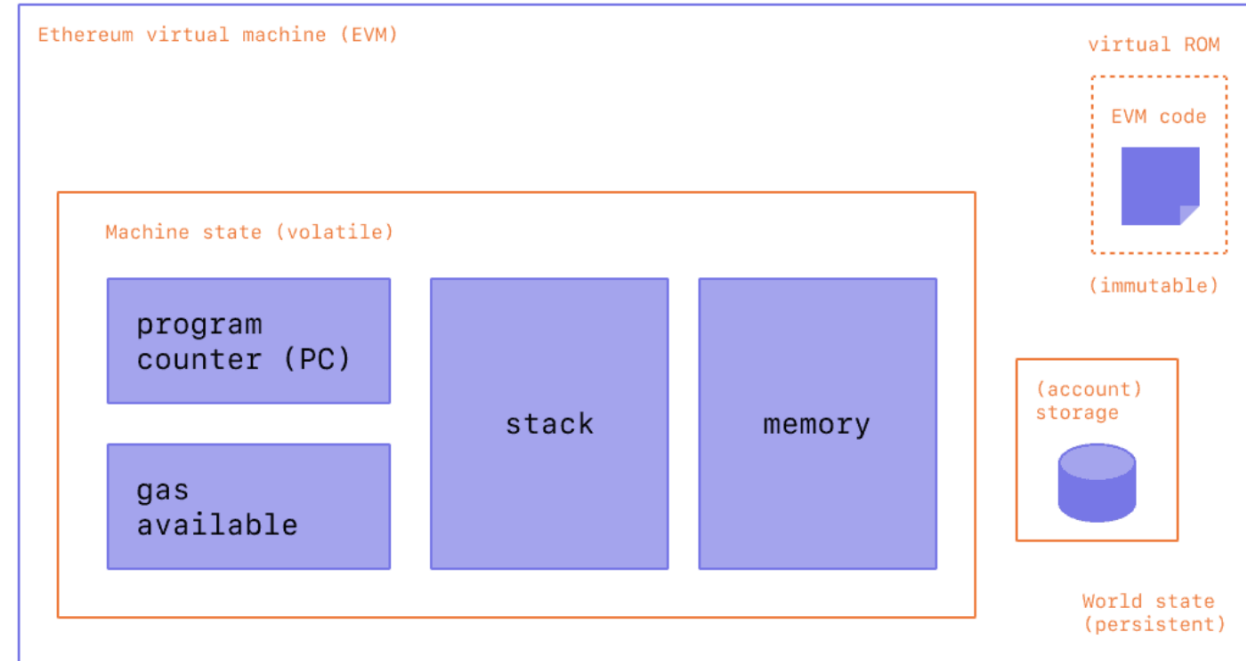
contract SimpleToken {
    mapping(address => uint256) public balanceOf;
    string public name;
    string public symbol;
    uint8 public decimals;
    uint256 public totalSupply;

    constructor() public {
        name = "Rohas Nagpal";
        symbol = "ROHAS";
        decimals = 18;
        totalSupply = 10000000000000000000;
        balanceOf[msg.sender] = totalSupply;
    }

    function transfer(address payable to, uint256 value) public {
        require(balanceOf[msg.sender] >= value && value > 0);
        balanceOf[msg.sender] -= value;
        balanceOf[to] += value;
    }
}
```

## EVM 구조, 동작과정

- ✨ *EVM*에 대한 보다 깊은 이해
- ✨ 블록체인 전반에 대한 이해
- ✨ 다른 체인을 이해할 때도 유리




## Foundry

- 🛠️ Ethereum 개발 및 테스트 도구
- 👍 유용함
  - `forge` : hardhat 처럼 씬
  - `cast` : 온체인과 소통할 때 씬
  - `anvil` : 로컬넷 돌리기



## Slither

-  Solidity 정적 분석 툴
- `slither <contract_code>.sol`  
하면 바로 결과 확인됨



```

🍏 > ~/aaron/test/slither_test
slither GatekeeperThree.sol
'solc --version' running
'solc GatekeeperThree.sol --combined-json abi,ast,bin,bin-runtime,srcmap,src
INFO:Detectors:
SimpleTrick.checkPassword(uint256) (GatekeeperThree.sol#13-19) uses a dangerous
    - _password == password (GatekeeperThree.sol#14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous
INFO:Detectors:
Modifier GatekeeperThree.gateThree() (GatekeeperThree.sol#54-58) does not
INFO:Detectors:
Reentrancy in GatekeeperThree.getAllowance(uint256) (GatekeeperThree.sol#60)
    External calls:
    - trick.checkPassword(_password) (GatekeeperThree.sol#61)
    State variables written after the call(s):
    - allowEntrance = true (GatekeeperThree.sol#62)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy
INFO:Detectors:

```



## 원래 목표

- 버그 바운티
  - 아직 못함;
- 구상
  - `slither` 를 이용해 정적분석으로 취약점 있는지 확인
  - 테스트넷, 로컬넷에서 공격 테스트
  - 문서로 만들어서 바운티 플랫폼에 제출

# 목표 선정

- 😊 제일 돈 적게 주는 곳

All🔍Hide All Filters (1) 🔼

Q Search by keyword

Filter By

Product Type

Ecosystem

Program Type

Project Type

Language 1

General

Select to limit results

☒ Solidity

☐ Rust

☐ JavaScript

☐ Go

☐ C/C++

☐ C#

☐ Clarity

Clear Filters

View 189 Bounties

Showing 189 bounty programs that match filtering criteria sorted by Max Bounty

https://github.com/yocashofficial/play

Assets in scope

https://github.com/yocashofficial/play

Target

Smart Contract - All smart contracts under / contracts in main branch

Type

Primacy Of Impact

Target

Smart Contract

Type

https://yoca.sh/

Target

Websites and Applications - Web Assets

Type

Primacy Of Impact

Target

Websites and Applications

Type

- 🔥 Slither 결과, 뭐가 많이 뜸
- 😊 Ethernaut 에서 연습한 공격 기법 적용하면 될 것 같다..

```
aarons-MacBook-Pro 1
~/aaron/test/play/contracts/Games > on main *1 !3 ?2
slither Common.sol
'solc --version' running
'solc Common.sol --combined-json abi,ast,bin,bin-runtime,srcmap,srcmap-runtime,userdoc,devdoc,hashes --allow-paths chat.openai.com/c/f0b806fb-782f-483f-be5f-fb1b371c2041' running
INFO:Detectors:
Common._transferWager(address,uint256,uint256,address) (Common.sol#85-124) uses arbitrary from in transferFrom
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#arbitrary-from-in-transferfrom
INFO:Detectors:
Common.ChainLinkVRF (Common.sol#62) is never initialized. It is used in:
- Common._requestRandomWords(uint32) (Common.sol#202-214)
Common.chainlinkVRFKeyHash (Common.sol#63) is never initialized. It is used in:
- Common._requestRandomWords(uint32) (Common.sol#202-214)
Common.chainlinkVRFSubscriptionId (Common.sol#64) is never initialized. It is used in:
- Common._requestRandomWords(uint32) (Common.sol#202-214)
Common.LINK_ETH_FEED (Common.sol#66) is never initialized. It is used in:
- Common.getVRFFee(uint256) (Common.sol#141-152)
Common.IChainLinkVRF (Common.sol#67) is never initialized. It is used in:
- Common.getVRFFee(uint256) (Common.sol#141-152)
Common.Bankroll (Common.sol#68) is never initialized. It is used in:
- Common._transferWager(address,uint256,uint256,address) (Common.sol#85-124)
- Common._transferToBankroll(address,uint256) (Common.sol#126-139)
- Common._transferPayout(address,uint256,address) (Common.sol#194-200)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
Common.getVRFFee(uint256) (Common.sol#141-152) ignores return value by (answer) = LINK_ETH_FEED.latestRoundData
Common.getVRFFee(uint256) (Common.sol#141-152) ignores return value by (fulfillmentFlatFeeLinkPPMTier1) = IChainLinkVRF.getVRFFee(uint256)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Common.transferFees(address).to (Common.sol#173) lacks a zero-check on :
- (success) = address(address(to)).call{value: fee}() (Common.sol#176)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

- 공격 성공하는거 확인하고, 문서화 한 다음 제출 예정



**Thank you**