Comparison of Smart Contract Vulnerability Detection tool

Member :

Kuo-Hao Lai kuohao1023@vt.edu
Alex Tsai alextsai1618@vt.edu
Md Mohaimin Al Barat barat@vt.edu

VIRGINIA TECH

Agenda

- Background
- Attack introduction
- Smart Contract Vulnerability analytic Tools
- Comparison of Vulnerability analytic Tools
- Conclusion
- Q&A

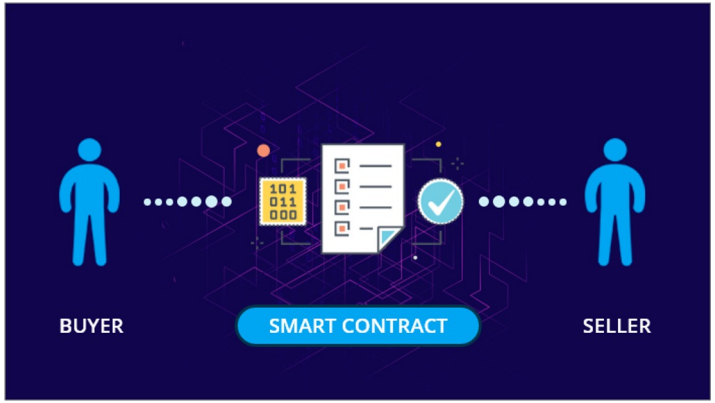
VIRGINIA TECH

Background

VIRGINIA TECH

Background

- Security challenges in decentralized systems
- Vulnerability detection techniques
- Best practices for smart contract security



VIRGINIA TECH

Security Challenges in Decentralized Systems

- Vulnerabilities in smart contract code
  - Exploit by attacker to steal funds
- Manipulation of oracle data
  - Incorrect data
  - Unintended behavior

VIRGINIA TECH

Security Challenges in Decentralized Systems (Con't)

- Risks of centralized points in decentralized systems
  - undermine the trust in the decentralized system
- Increased targeting by bad actors
  - harm the reputation of the entire ecosystem
- Example:
  - DAO attack
  - DeFi flash loan attacks
  - Centralized exchange hacks

VIRGINIA TECH

Best Practices for Smart Contract Security

- Conduct thorough audits
  - prevent reentrancy attacks
- Use established coding patterns
  - secure access control
- Limit contract complexity
  - mitigate front-running attacks
- Implement upgradeability and emergency stop mechanisms
  - prevent potential vulnerabilities

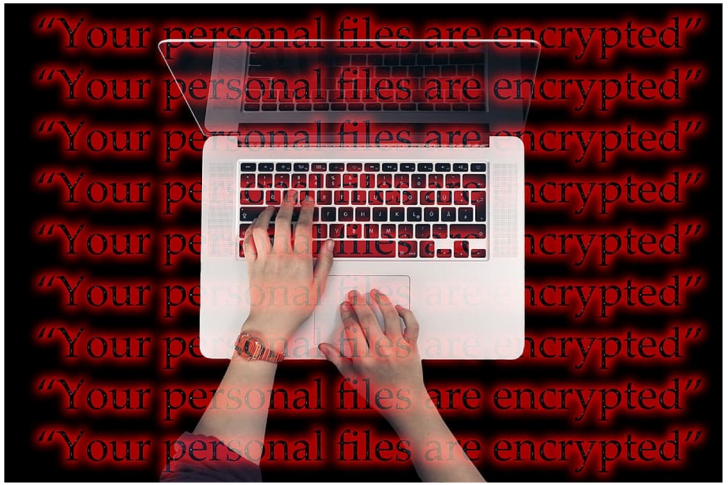
VIRGINIA TECH

Attack Introduction

VIRGINIA TECH

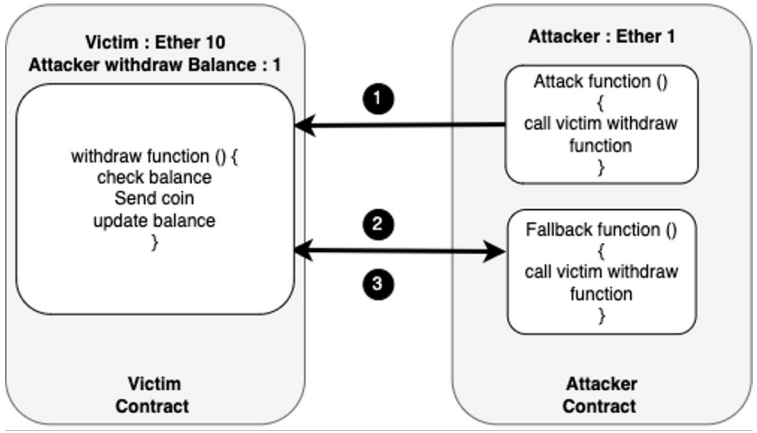
Attack Introduction

- Reentrance attacks
- Access Control attacks
- Front Running attacks
- Unchecked low-level calls attacks
- Importance of security measures



VIRGINIA TECH

Reentrance attacks



Access Control attacks

```
1   pragma solidity ^0.6.0;
2
3   contract Wallet {
4       address public owner;
5       mapping(address => uint) public balances;
6
7       constructor() public {
8           owner = msg.sender;
9       }
10
11      function deposit() public payable {
12          balances[msg.sender] += msg.value;
13      }
14
15      function withdraw(uint amount) public {
16          require(balances[msg.sender] >= amount, "Insufficient balance.");
17          msg.sender.call{value: amount}("");
18          balances[msg.sender] -= amount;
19      }
20
21      function transferOwnership(address newOwner) public {
22          owner = newOwner;
23      }
24  }
```

VIRGINIA TECH

Front Running attacks

```
1   // Original Source Code
2   contract KingOfEtherThrone {
3       address public king;
4       uint public balance;
5
6       function claimThrone() public payable {
7           require(
8               msg.value > balance,
9               "Insufficient payment to claim the throne."
10          );
11          king.transfer(balance);
12          king = msg.sender;
13          balance = msg.value;
14      }
15  }
```

VIRGINIA TECH

Unchecked low-level calls attacks

```
1   pragma solidity ^0.6.0;
2
3   contract UncheckedCalls {
4       mapping(address => uint) public balances;
5
6       function deposit() public payable {
7           balances[msg.sender] += msg.value;
8       }
9
10      function withdraw(uint amount) public {
11          require(balances[msg.sender] >= amount, "Insufficient balance.");
12
13          (bool success,) = msg.sender.call{value: amount}("");
14          if (!success) {
15              revert("Withdrawal failed.");
16          }
17
18          balances[msg.sender] -= amount;
19      }
20  }
```

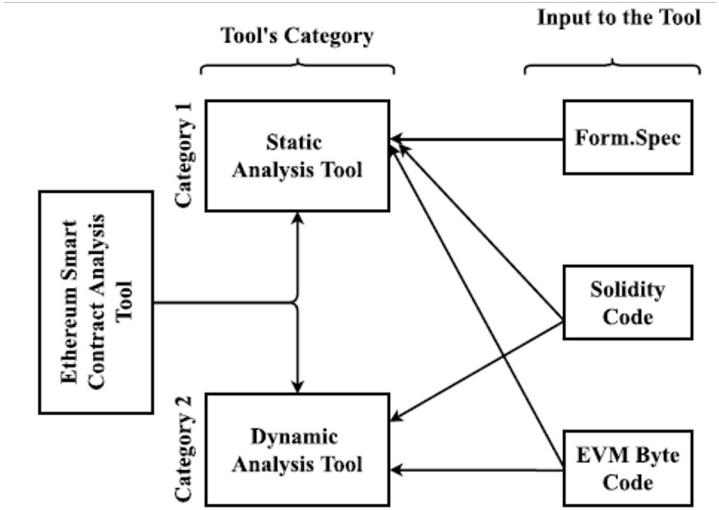
VIRGINIA TECH

Smart Contract Vulnerability analytic Tool

VIRGINIA TECH

Analysis Tools

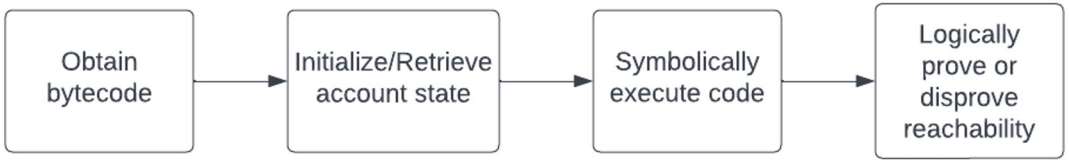
- Why do we need?
- Categories:
  - Static Analysis Tool
    - Mythril, Smartcheck, etc.
  - Dynamic Analysis Tool
    - contractLarva, MAIAN etc.



VIRGINIA TECH

Analysis Tools: Mythril

- Analyzes smart contracts written with Solidity
- Takes advantage of the symbolic execution technique with taint analysis
- Steps:



VIRGINIA TECH

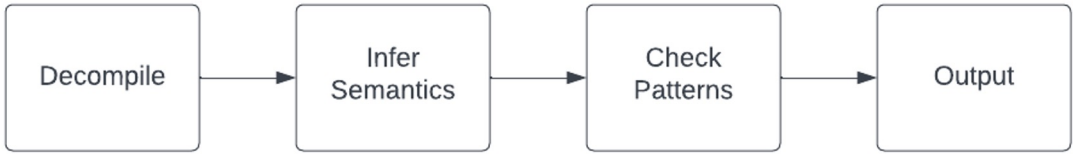
Analysis Tools: Mythril

- Limitations
  - Unable to extend taints over memory fields
  - Becomes worse when the parameters accept pass by reference

VIRGINIA TECH

Analysis Tools: Securify

- Statically analyzes EVM bytecode
- Checks compliance and violation patterns
- Steps:



VIRGINIA TECH

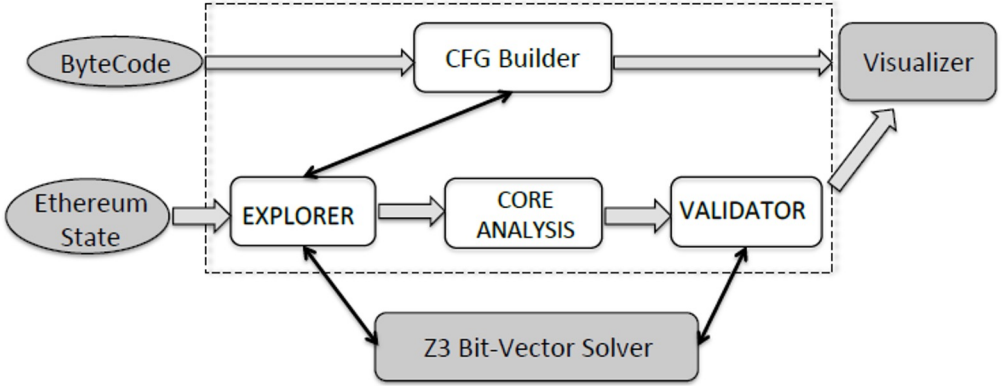
Analysis Tools: Securify

- Limitations:
  - Cannot reason about numerical properties
  - Does not reason about reachability
  - Can be exploited by attackers

VIRGINIA TECH

Analysis Tools: Oyente

- Based on symbolic execution
- Architectural overview:



VIRGINIA TECH

Analysis Tools: Oyente

- Limitations:
  - Fails to log 72.9% of the TOD
  - Detects very few vulnerabilities
  - Generates false positives
  - Underestimates some serious bugs

VIRGINIA TECH

Analysis Tools: SmartCheck

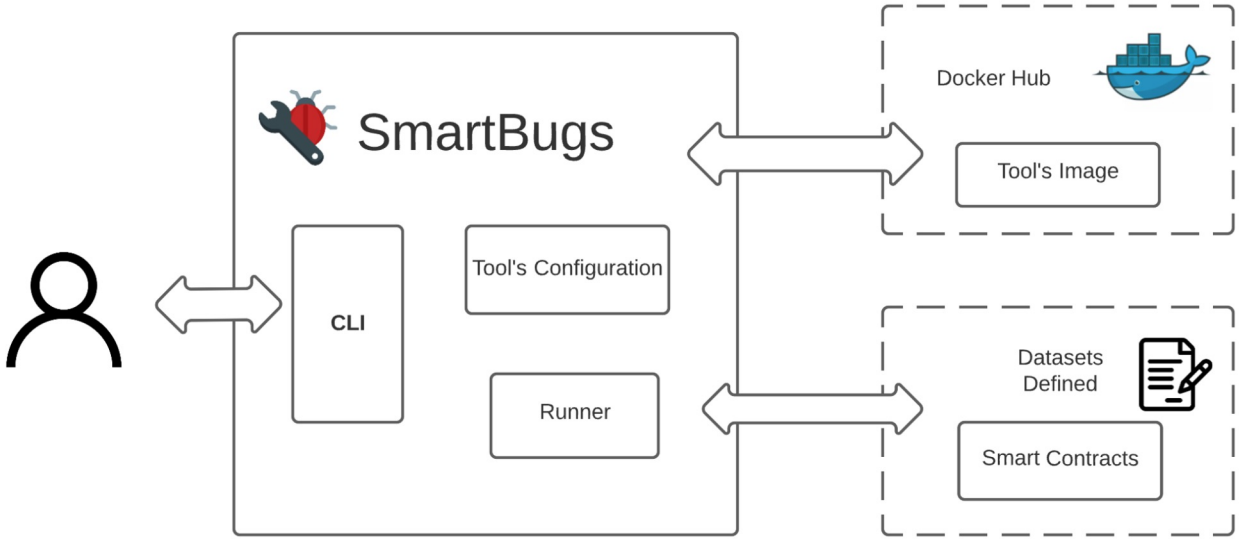
- Runs lexical and syntactical analysis
- Uses XML file with source code in tree form
- Explores path that can lead to vulnerabilities
- Detects patterns by using XPath queries
- Limitations:
  - Unable to detect vulnerabilities with taint analysis
  - Unable to detect Front Running
  - Only identifies low risk vulnerabilities

VIRGINIA TECH

Comparison of Vulnerability analytic Tools

VIRGINIA TECH

SmartBug Architecture



VIRGINIA TECH

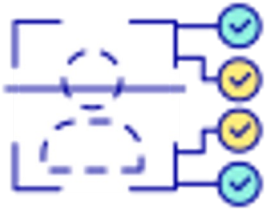
Dataset

- $SB^{CURATED}$:
  - Consists of 69 vulnerable smart contracts.
- SB^{WILD}:
  - Contains 47,518 contracts extracted from the Ethereum blockchain

VIRGINIA TECH

Evaluation metrics

- Accuracy
- Number of Detected Vulnerabilities
- Execution Time



VIRGINIA TECH

Result

Accuracy

Category	Mythril	Oyente	Securify	Smartcheck	Total
Access Control	4/19 21%	0/19 0%	0/19 0%	2/19 11%	4/19 21%
Arithmetic	15/22 68%	12/22 55%	0/22 0%	1/22 5%	17/22 77%
Denial Service	0/7 0%	0/7 0%	0/7 0%	0/7 0%	0/ 7 0%
Front Running	2/7 29%	0/7 0%	2/7 29%	0/7 0%	2/ 7 29%
Reentrancy	5/8 62%	5/8 62%	5/8 62%	5/8 62%	5/ 8 62%
Time Manipulation	0/5 0%	0/5 0%	0/5 0%	1/5 20%	1/ 5 20%
Unchecked Low Calls	5/12 42%	0/12 0%	3/12 25%	4/12 33%	6/12 50%
Other	0/3 0%	0/3 0%	0/3 0%	0/3 0%	0/ 3 0%
Total	31/115 27%	17/115 15%	10/115 9%	13/115 11%	35/115 30%

Accuracy

VIRGINIA TECH

Result

Combine tools

	Mythril	Oyente	Securify	Smartcheck
Mythril		33/115 29%	31/115 27%	33/115 29%
Oyente			22/115 19%	25/115 22%
Securify				16/115 14%
Smartcheck				

Accuracy from combining tools

VIRGINIA TECH

Result

Category	Mythril	Oyente	Securify	Smartcheck
Access Control	1076 2%	2 0%	614 1%	384 0%
Arithmetic	18,515 39%	34,306 72%	0 0%	7,430 15%
Denial Service	0 0%	880 1%	0 0%	11,621 24%
Front Running	2,015 4%	0 0%	7,217 15%	0 0%
Reentrancy	8,454 17%	308 0%	2,033 4%	847 1%
Time Manipulation	443 0%	0 0%	592 1%	2,867 6%
Unchecked Low Calls	443 0	0 0%	592 1%	2,867 6%
Total	22,994 48%	34,764 73%	8,781 18%	24,906 52%

false positives!!

Vulnerabilities

Numbers of Contracts that have at least one vulnerability

VIRGINIA TECH

Result

Tools	Average	Total
Mythril	0:01:24	46 days, 07:46:55
Oyente	0:00:30	16 days, 04:50:11
Securify	0:06:37	217 days, 22:46:26
SmartCheck	0:00:10	5 days, 12:33:14
Total	0:01:40	330 days and 15 hours

VIRGINIA TECH

Conclusion

VIRGINIA TECH

Conclusion

- Accuracy: Mythril 
- Number of Detected Vulnerabilities: Oyente 
- Execution Time: Oyente and SmartCheck 

VIRGINIA TECH

Conclusion

- Comparison of four security analysis tools:
  - Oyente
  - Mythril
  - Securify
  - Smartcheck
- SmartBugs
- Future Work

VIRGINIA TECH

Q&A

VIRGINIA TECH