A Mixed-Methods Study of Security Practices of Smart Contract Developers




Tanusree Sharma
UIUC

Zhixuan Zhou
UIUC

Andrew Miller
UIUC

Yang Wang
UIUC

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Motivation

Long-term goal: design tools to identify and mitigate smart contract vulnerabilities

This study: understand how smart contract developers currently deal with security

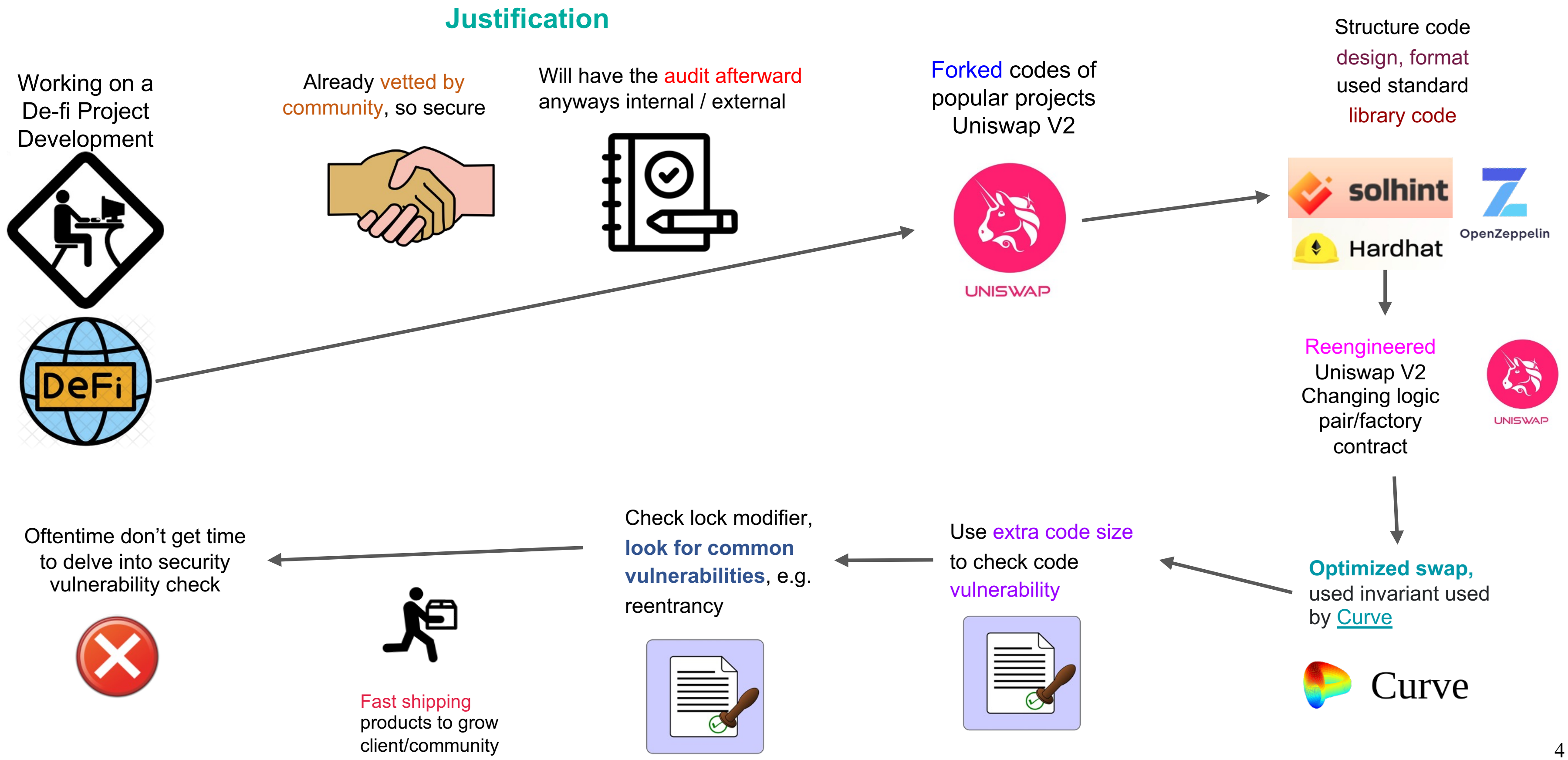
Code with Reentrancy

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.10;
 contract Dao {
    mapping(address => uint256) public balances;
    function deposit() public payable {
        require(msg.value >= 1 ether, "Deposits must be no
less than 1 Ether");
        balances[msg.sender] += msg.value;
    }
    function withdraw() public {
        // Check user's balance
        require(
            balances[msg.sender] >= 1 ether,
            "Insufficient funds.  Cannot withdraw"
        );
        uint256 bal = balances[msg.sender];
        // Withdraw user's balance
        (bool sent, ) = msg.sender.call{value: bal}("");
        require(sent, "Failed to withdraw sender's balance");
        // Update user's balance.
        balances[msg.sender] = 0;
    }
    function daoBalance() public view returns (uint256) {
        return address(this).balance;
    }
}
```

Fixing Reentrancy

```
Contract Dao {
…

     function withdraw() public {
        // Check user's balance
        require(
            balances[msg.sender] >= 1 ether,
            "Insufficient funds.  Cannot withdraw"
        );
        uint256 bal = balances[msg.sender];

        // Update user's balance.
        balances[msg.sender] = 0;

        // Withdraw user's balance
        (bool sent, ) = msg.sender.call{value:
bal}("");
        require(sent, "Failed to withdraw sender's
balance");

        // Update user's balance.
        balances[msg.sender] = 0;
    }
```

A Developer Journey (P1)



Research Questions

RQ1 How do smart contract developers ensure their smart contracts are secure against potential attacks?

RQ2 How do smart contract developers conduct code reviews and whether they are able to identify common smart contract security vulnerabilities in the code?

Interview + Code Review

We conducted an interview and code review session with 29 Smart Contract Developers from 10 countries

Participants
Users

Recruitment
Mailing list

Exploratory Interview
Experience & current practices

→ **Code Review**
To identify vulnerabilities

→ **Exit Interview**
Experience of code review

Gender

24	Male
5	Female

Geography

1	USA
1	India
5	Canada
2	Australia
2	Germany
2	New Zealand
1	Greece
1	China
1	Egypt
1	Ghana
1	Iran
1	UK
1	

Years of Experience

10	<1 years
10	1-3 years
9	+3 years

Occupation

14	Full-time DeFi smart contract Developer	
3	Freelance smart contract Develope	
8	Smart contract Developer-student	
3	Software developer different domain	
1	Professor	6

Gender

69%	Male
31%	Female

Occupation

79%	Full-time DeFi smart contract Developer
44%	Smart Contract Protocol development
35%	Smart Contract Development
13%	Smart Contract Research/security assessment

Years of Experience

7.6%	<1 years
25.7%	1-3 years
66.7%	+3 years

Survey + Code Review

We conducted online survey with **171** Smart Contract Developers

Survey
Current Experience & Practice

Code Review
To identify vulnerabilities

Results

- Security Perceptions
- Security Practices
- Security Behaviors

“Security was not a priority”

“If you’re planning to do an audit anyway, it kind of makes sense from a business perspective to ship code and then run it through multiple audits, instead of having your internal team [...]review the security at the same time.”

- P8

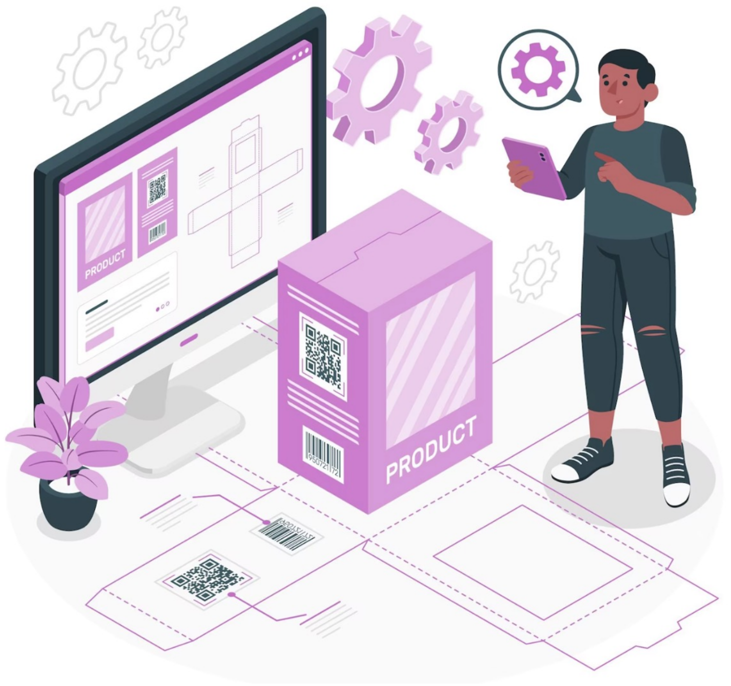


Image source: https://www.flaticon.com/

“Smart of Contract Security is Hard”

``Contract work[s] like state machine when send a transaction. It only appears like state changes. But in regular program, you can differentiate read-only calls and state changes. Solidity can not do that.'' – ***P19***

Image source: https://www.flaticon.com/

Developers had broadly 3 common practices for security in smart contracts

Smart Contract Security Practices

Software engineering best practices

Importance of code refactoring & using vetted libraries

“write the most simple code that you can and draw the diagram to visualize the flow of smart contract code design” - P20

Common software testing techniques

Code reviews, input validations, and static analyses

“Having internal team for code review… in this culture of moving fast and breaking things. Also audits from external entities. - P10’’

Specialized strategies

Creating own bytecode dictionary

“I created own bytecode (error code) dictionary to represent different cases of reverting transactions in his smart contracts for an NFT (non-fungible token) project - P18.’’

Frequently used **Truffle testing suite, Remi, Hardhat, Slither, MythX**

Existing **symbolic execution based tools**, are limited in identifying edge case,

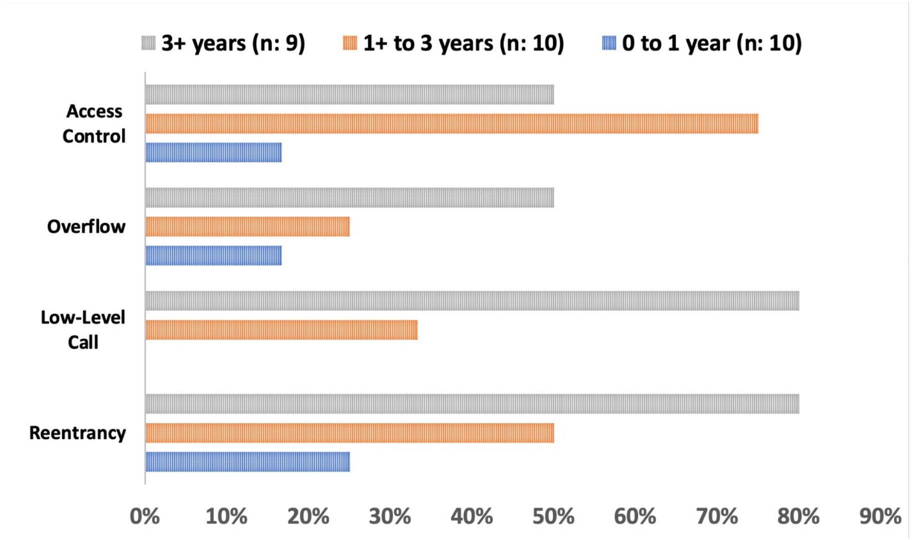
Use of Security Tooling, Limitations of smart contract security tools, & Code Review Practice

Manual inspection (64%) was frequently used method for smart contract security

Developers Security Practices in Action

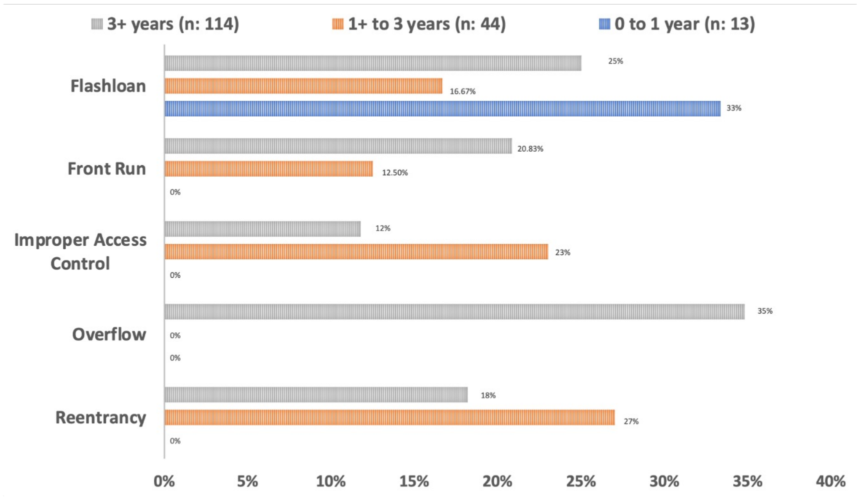
Code Review Result - Interview

Overall, 55% of (16 out of 29) identify one or more vulnerabilities. 28% (N=8) of identified both (all) vulnerabilities

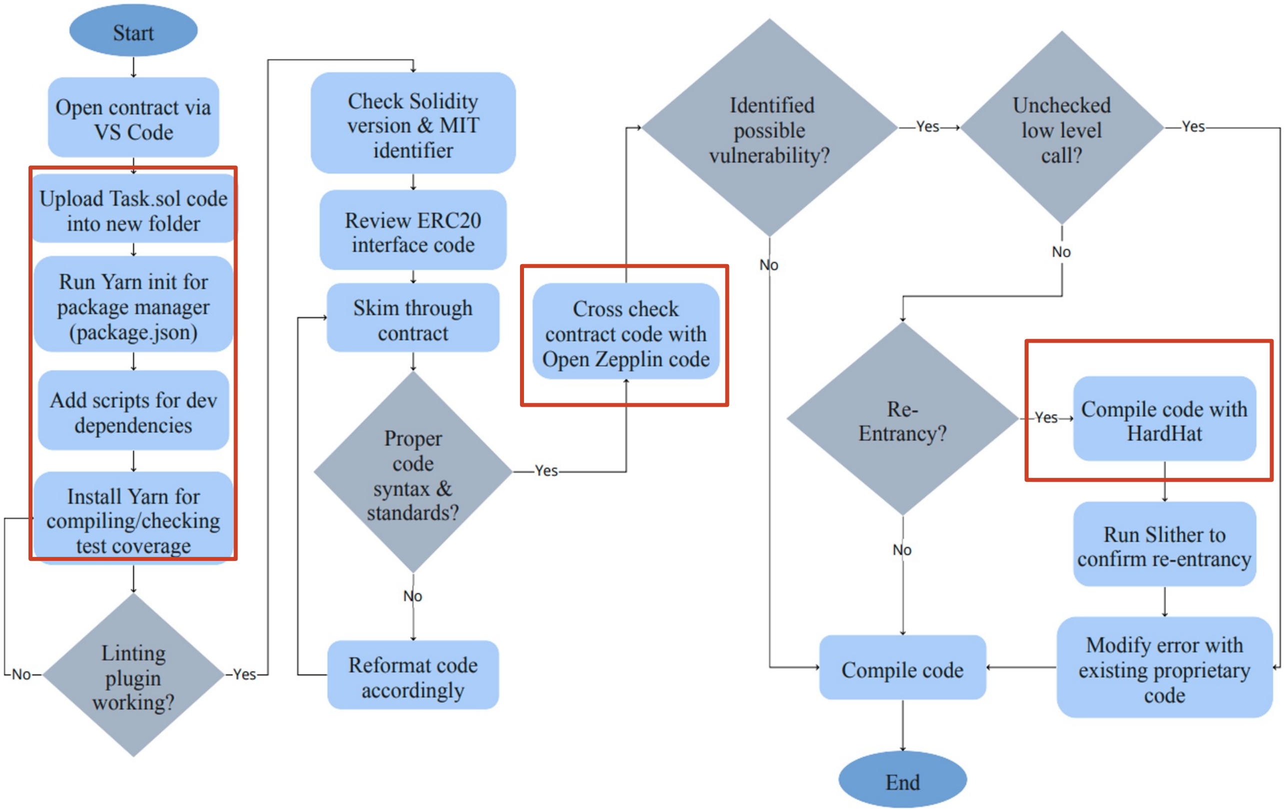


Survey

20.5% (n: 171) identified vulnerability.



Smart Contract Security Practice in Action



"it is withdrawing if the amount is less than the amount to just return false and subtracts the amount before it does the accounting before it's sending anything out, which is pretty crucial for preventing someone re-entering the function, which would be bad. - P14"

Design implications

Education & Standards

Hands-on exercises or labs, incorporate education teachable moments in Compilers, Security tools, IDEs, Testnets

Image source: https://www.flaticon.com/

Design implications

User interfaces & user experience

Actionable insights through Error / warning messages - zooming into where exactly the problems are in the code and how significant the effect can be

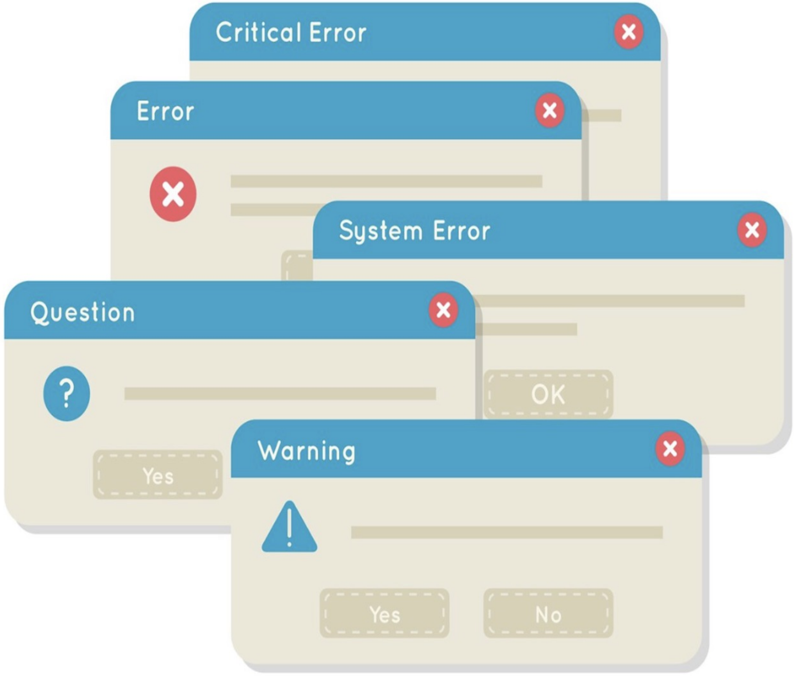


Image source: https://www.flaticon.com/

Thank you!

Contact:
tsharma6@illinois.edu

@Tanusree_Sharma

Paper QR Code:

Key Takeaways

- ❖ **Limitation in tooling**
  - ➢ Tailored Education, Standards, hands on Lab based on experience level
  - ➢ Hierarchical and self explainable Error Message in security/ development platform
  - ➢ Comprehensibility of Code libraries, symbolic execution tooling
- ❖ **Future Research can explore**
  - ➢ Impact of Smart Contract Development Culture's impact on security
  - ➢ Comparison study with developers of different smart contracts language (e.g. solidity, vyper, etc)

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN