Using My Functions Should Follow My Checks: Understanding and Detecting Insecure OpenZeppelin Code in Smart Contracts

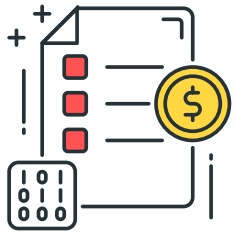
Han Liu[1], DaoyuanWu[2], **Yuqiang Sun[3]**, Haijun Wang[4], Kaixuan Li[1], Yang Liu[3], Yixiang Chen[1]

[1] East China Normal University
[2] The Hong Kong University of Science and Technology
[3] Nanyang Technological University
[4] Xi'an Jiaotong University

suny0056@e.ntu.edu.sg

19 August 2024

Smart Contracts



- Smart contracts are programs running on blockchains.
- They usually provide **financial services**.
- Attacks on smart contracts has caused more than **$1,000,000,000** loss.

SAST for Smart Contracts

- Most tools are rule-based.
- **80%** of vulnerabilities are machine undetectable.
- Rules are patterns for **insecure** implementation.

If we could learn from the secure implementations?

2 Motivating Example

Solidity version < 0.8.0

```
1  function flashLoan(
2      IERC3156FlashBorrowerUpgradeable receiver,
3      address token,
4      uint256 amount,
5      bytes memory data
6  ) public virtual override returns (bool)
7  {
8      // Vulnerable point: It misses the amount check in the
       original OpenZeppelin library; see Line 7 in Figure 6.
9      uint256 fee = flashFee(token, amount);
10     _mint(address(receiver), amount);
11     require(receiver.onFlashLoan(msg.sender, token, amount, fee,
       data) == RETURN_VALUE, "ERC20FlashMint: invalid return value")
       ;
12     uint256 currentAllowance = allowance(address(receiver),
       address(this));
13     require(currentAllowance >= amount + fee, "ERC20FlashMint:
       allowance does not allow refund");
14     _approve(address(receiver), address(this),
       amount - fee);
15     _burn(address(receiver), amount + fee);
16     return true;
17 }
```

Amount + fee overflow

Vulnerable contracts NFTX from the Code4rena audit report

```
1  function maxFlashLoan(address token) public view virtual returns
       (uint256) {
2      return token == address(this) ? type(uint256).max -
       totalSupply() : 0;
3  }
4  function flashLoan(
5      IERC3156FlashBorrower receiver,
6      address token,
7      uint256 amount,
8      bytes calldata data
9  ) public virtual override returns (bool) {
10     require(amount <= maxFlashLoan(token), "ERC20FlashMint:
       amount exceeds maxFlashLoan");
11     uint256 fee = flashFee(token, amount);
12     _mint(address(receiver), amount);
13     require(
14         receiver.onFlashLoan(msg.sender, token, amount, fee, data
       ) == _RETURN_VALUE,
15         "ERC20FlashMint: invalid return value"
16     );
17     address flashFeeReceiver = _flashFeeReceiver();
18     _spendAllowance(address(receiver), address(this), amount +
       fee);
19     ...
20 }
```

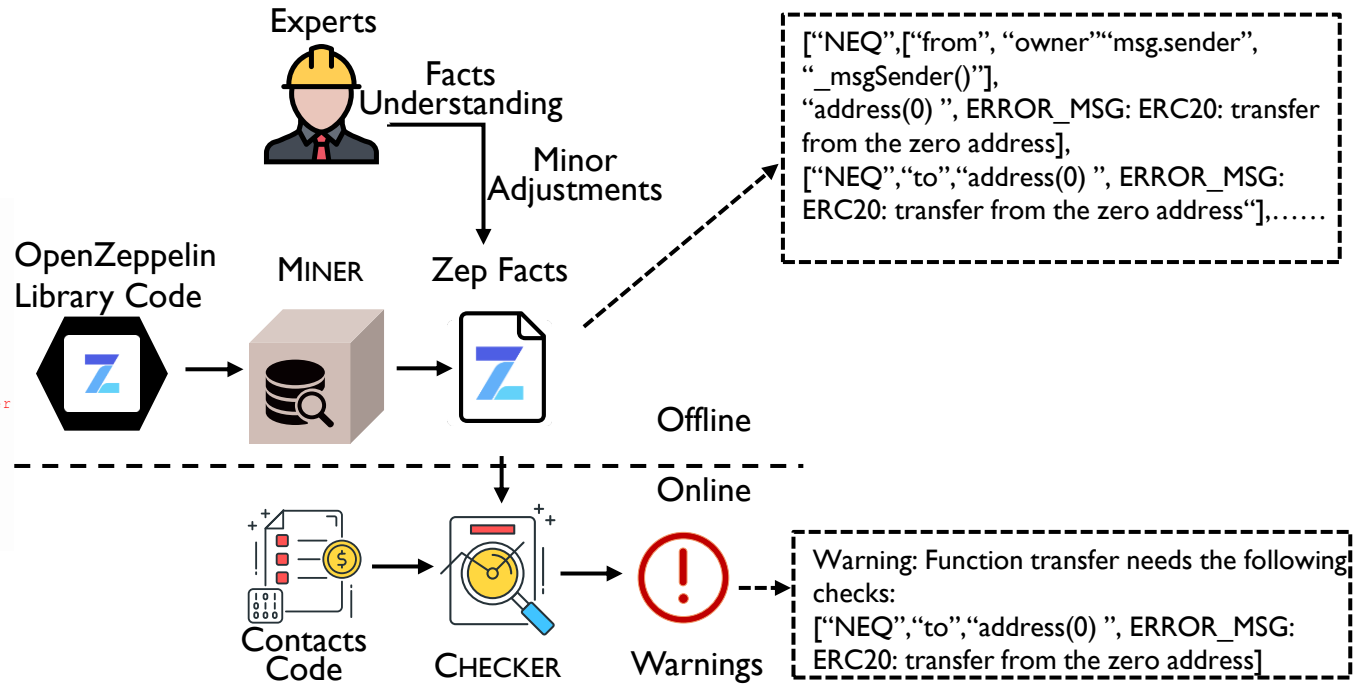
OpenZeppelin

Protection in OpenZeppelin

Standard contracts in OpenZeppelin Code

3 Framework of ZepScope

```
1  function transfer(address to, uint256 amount)
       public virtual override returns (bool) {
2      address owner = _msgSender();
3      _transfer(owner, to, amount);
4      return true;
5  }
6  function _transfer(address from, address to,
       uint256 amount) internal virtual {
7      require(from != address(0), "ERC20:
           transfer from the zero address");
8      require(to != address(0), "ERC20: transfer
           to the zero address");
9      _beforeTokenTransfer(from, to, amount);
10     uint256 fromBalance = _balances[from];
11     require(fromBalance >= amount, "ERC20:
           transfer amount exceeds balance");
12     ...
13 }
```



The Framework of ZepScope

3 Framework of ZepScope

Mining OpenZeppelin Function Check

Fact **before** function call (*_mint*)

Fact **during** function call (*_mint*)

Fact **after** function call (*_mint*)

```
function _mint(address account, uint256 amount) internal virtual {
    require(account != address(0), "ERC20: mint to the zero address");
    _beforeTokenTransfer(address(0), account, amount);
    _totalSupply += amount;
    _balances[account] += amount;
    emit Transfer(address(0), account, amount);
    _afterTokenTransfer(address(0), account, amount);
}
function flashLoan(
    IERC3156FlashBorrower receiver,
    address token,
    uint256 amount,
    bytes calldata data
) public virtual override returns (bool) {
    require(amount <= maxFlashLoan(token), "ERC20FlashMint: amount exceeds maxFlashLoan");
    uint256 fee = flashFee(token, amount);
    _mint(address(receiver), amount);
    require(
        receiver.onFlashLoan(msg.sender, token, amount, fee, data) == _RETURN_VALUE,
        "ERC20FlashMint: invalid return value"
    );
    address flashFeeReceiver = _flashFeeReceiver();
    _spendAllowance(address(receiver), address(this), amount + fee);
    ...
}
```

3 Framework of ZepScope

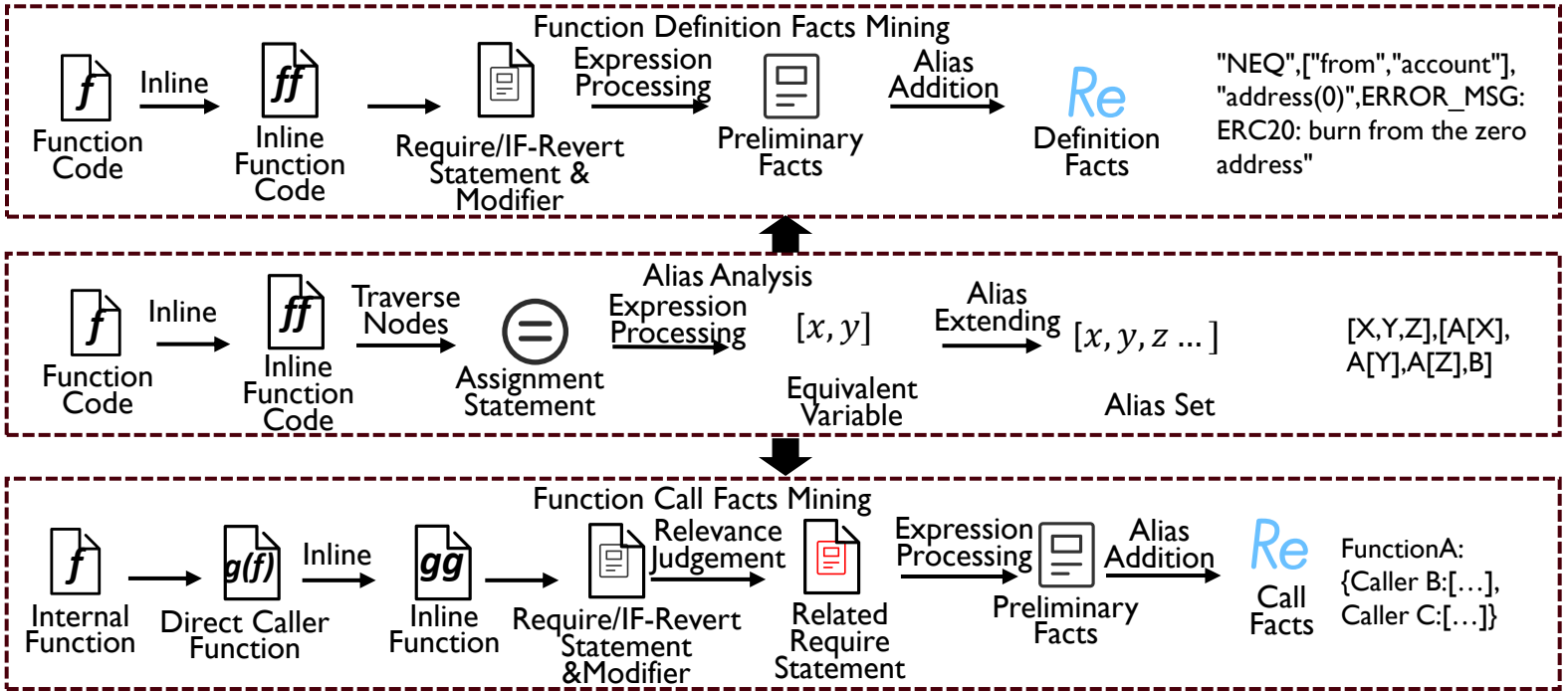
Challenge in Mining OpenZeppelin Function Check

- Alias analysis needed
- Relevance between *facts after function call* and the *function call* itself.

```
1  function flashLoan(
2      IERC3156FlashBorrower receiver,
3      address token,
4      uint256 amount,
5      bytes calldata data
6  ) public virtual override returns (bool) {
7      require(amount <= maxFlashLoan(token), "ERC20FlashMint:
       amount exceeds maxFlashLoan");
8      uint256 fee = flashFee(token, amount);
9      _mint(address(receiver), amount);
10     require(
11         receiver.onFlashLoan(msg.sender, token, amount, fee, data
       ) == _RETURN_VALUE,
12         "ERC20FlashMint: invalid return value"
13     );
14     address flashFeeReceiver = _flashFeeReceiver();
15     _spendAllowance(address(receiver), address(this), amount +
       fee);
16     ...
17 }
```

3 Framework of ZepScope

Workflow of MINER



3 Framework of ZepScope

Understanding OpenZeppelin Facts

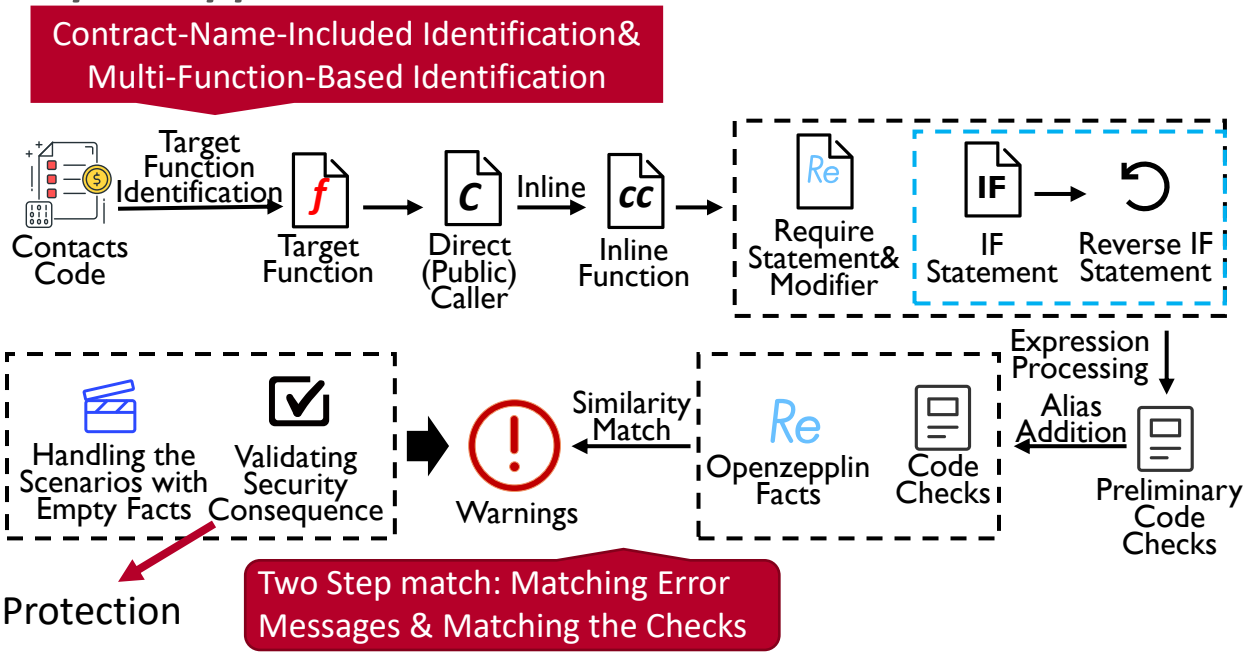
Total 1,435 facts, divided into four major categories:

- Address Compliance Assurance
- Access Control
- Overflow/Underflow Check
- Timestamp or State Check

277 high-level, 858 medium-level, and 300 low-level facts

3 Framework of ZepScope

Detecting Insecure OpenZeppelin Code in SCs



- Equivalent Overflow Protection
- Equivalent Permissions
- Extra msg.value Checks

The workflow of CHECKER

4 Evaluation

RQ1: Comparison with the SOTA Tools

Datasets:

51 real-world security bugs caused by insecure OpenZeppelin code. These bugs were sourced from security incidents reported on *DeFiHackLabs, Twitter, SmartBugs Curated datasets, and audit reports from Code4rena, Sherlock, and Ethereum Commonwealth*

Tool	TP	FP	FN	# Failed
Slither	8	32	43	0
AChecker	0	0	43	8
SoMo (via MetaScan)	8	22	43	0
ZepScope	41	0	10	0

4 Evaluation

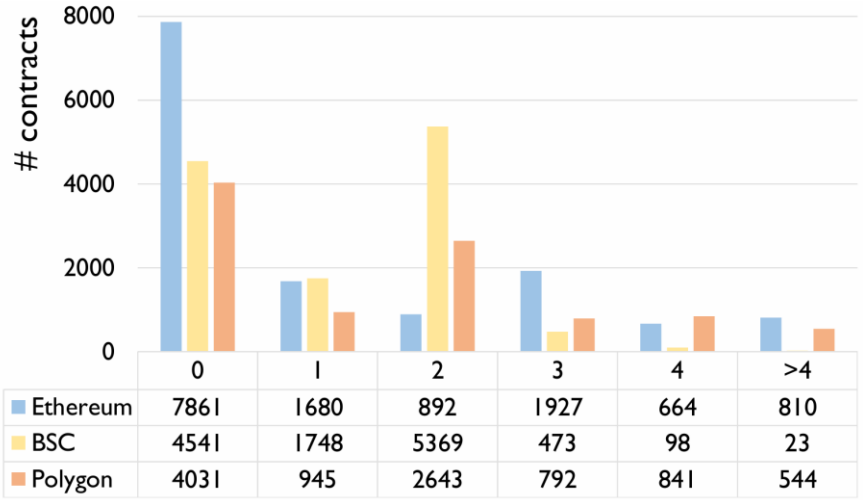
RQ2: Accuracy and Performance

Datasets:

top 15,000 contracts of three chains (Ethereum, BSC, and Polygon), ranked by the balances of the contracts

Chain	# Contacts	# Failed	# Functions	# Warnings	Sampled Accuracy
Ethereum	13,984	150	911,309	16,873	84%
BSC	12,486	234	1,068,035	14,444	95%
Polygon	9,955	159	770,821	16,114	90%
Total	36,425	543	2,750,165	47,431	89.67%

Results



Warning Distribution

Performance : 42.39 seconds on average per contract

4 Evaluation

RQ3: Security Findings

Finding 1: 15 new vulnerabilities involves contracts contain $439,333+

```
function transferOwnership(address newOwner)
    public virtual {
    require(newOwner != address(0), "Ownable:
        new owner is the zero address");
    emit OwnershipTransferred(owner, newOwner);
    owner = newOwner;
}
```

```
function burn(uint _id) public {
    _burn(_id);
}
function _burn(uint256 tokenId) internal
    virtual {
    address owner = ERC721.ownerOf(tokenId);
    _beforeTokenTransfer(owner, address(0),
        tokenId);
    owner = ERC721.ownerOf(tokenId);
    delete _tokenApprovals[tokenId];
    unchecked {_balances[owner] -= 1;}
    delete _owners[tokenId];
    emit Transfer(owner, address(0), tokenId);
    _afterTokenTransfer(owner, address(0),
        tokenId);
}
```

4 Evaluation

RQ3: Security Findings

Finding 2: Pervasive Absence of Zero Address Checks

```
1  function transfer(address to, uint256 amount)
       public virtual override returns (bool) {
2      address owner = _msgSender();
3      _transfer(owner, to, amount);
4      return true;
5  }
6  function _transfer(address from, address to,
       uint256 amount) internal virtual {
7      require(from != address(0), "ERC20:
           transfer from the zero address");
8      require(to != address(0), "ERC20: transfer
           to the zero address");
9      _beforeTokenTransfer(from, to, amount);
10     uint256 fromBalance = _balances[from];
11     require(fromBalance >= amount, "ERC20:
           transfer amount exceeds balance");
12     ...
13 }
```

- Avoid unintentional permanent locking of tokens due to human errors or software glitches
- Differentiate the _transfer function from the burn function
- Avoid inaccuracies in the total supply figures while also preventing extra gas fee loss

! Can lead phishing attacks

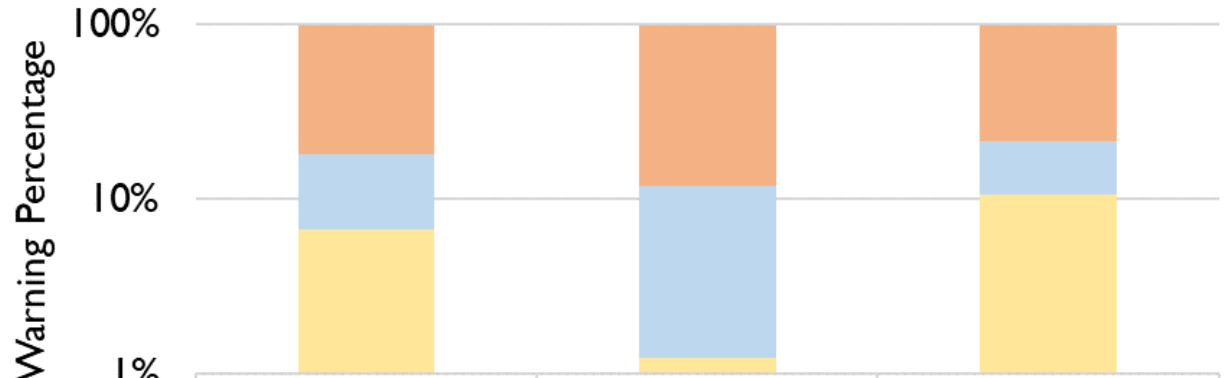
RQ3: Security Findings

Finding 3: A Campaign of Intentionally Loosing the Checks

```
1  function buy(address _refer) payable public
       returns(bool){
2      require(_swSale && block.number <=
           saleMaxBlock,"Transaction recovery");
3      require(msg.value >= 0.01 ether,"
           Transaction recovery");
4      uint256 _msgValue = msg.value;
5      uint256 _token = _msgValue.mul(salePrice);
6      _mint(_msgSender(),_token);
7      if(_msgSender()!=_refer&&_refer!=address(0)
           &&_balances[_refer]>0){
8          uint referToken = _token.mul(
               _referToken).div(10000);
9          uint referEth = _msgValue.mul(_referEth
               ).div(10000);
10         _mint(_refer,referToken);
11         address(uint160(_refer)).transfer(
               referEth);
12     }
13     return true;
14 }
```

4 Evaluation

RQ4: Cross-Chain Result Comparison



	Ethereum	BSC	Polygon
Low	82.04%	88.15%	78.70%
Medium	11.28%	10.63%	10.69%
High	6.69%	1.23%	10.61%

Thanks & QA

Han Liu[1], DaoyuanWu[2], **Yuqiang Sun[3]**, Haijun Wang[4], Kaixuan Li[1], Yang Liu[3], Yixiang Chen[1]

[1] East China Normal University
[2] The Hong Kong University of Science and Technology
[3] Nanyang Technological University
[4] Xi'an Jiaotong University

Email: suny0056@e.ntu.edu.sg