



ECNU



香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY



NANYANG
TECHNOLOGICAL
UNIVERSITY
SINGAPORE



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

Han Liu¹, DaoyuanWu², Yuqiang Sun³, Haijun Wang⁴, Kaixuan Li¹, Yang Liu³, Yixiang Chen¹

¹ East China Normal University

² The Hong Kong University of Science and Technology

³ Nanyang Technological University

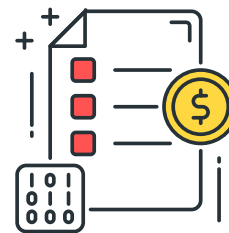
⁴ Xi'an Jiaotong University

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  
9     // original OpenZeppelin library; see Line 7 in Figure 6.  
10    uint256 fee = flashFee(token, amount);  
11    _mint(address(receiver), amount);  
12    require(receiver.onFlashLoan(msg.sender, token, amount, fee,  
13    data) == RETURN_VALUE, "ERC20FlashMint: invalid return value");  
14    ;  
15    uint256 currentAllowance = allowance(address(receiver),  
16    address(this));  
17    require(currentAllowance >= amount + fee, "ERC20FlashMint:  
18    allowance does not allow refund");  
19    _approve(address(receiver), address(this),  
20    amount - fee);  
21    _burn(address(receiver), amount + fee);  
22    return true;  
23 }
```

Vulnerable contracts NFTX
from the Code4rena audit report

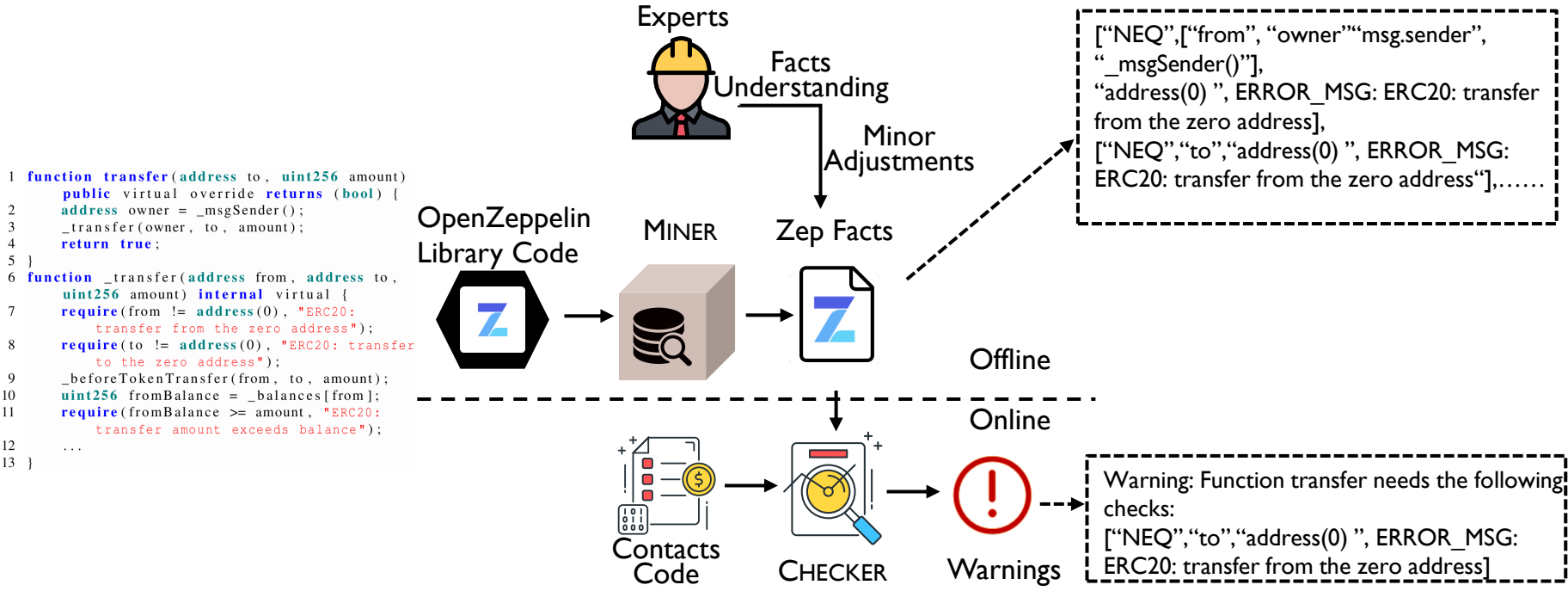
```
1 function maxFlashLoan(address token) public view virtual returns  
2     (uint256) {  
3     return token == address(this) ? type(uint256).max -  
4     totalSupply() : 0;  
5 }  
6 function flashLoan(  
7     IERC3156FlashBorrower receiver,  
8     address token,  
9     uint256 amount,  
10    bytes calldata data  
11 ) public virtual override returns (bool) {  
12    require(amount <= maxFlashLoan(token), "ERC20FlashMint:  
13    amount exceeds maxFlashLoan");  
14    uint256 fee = flashFee(token, amount);  
15    _mint(address(receiver), amount);  
16    require(  
17        receiver.onFlashLoan(msg.sender, token, amount, fee, data  
18    ) == _RETURN_VALUE,  
19        "ERC20FlashMint: invalid return value"  
20    );  
21    address flashFeeReceiver = _flashFeeReceiver();  
22    _spendAllowance(address(receiver), address(this), amount +  
23    fee);  
24    ...  
25 }
```

 OpenZeppelin

Protection in
OpenZeppelin

Standard contracts in
OpenZeppelin Code

3 Framework of ZepScope



```

1 function transfer(address to, uint256 amount)
2   public virtual override returns (bool) {
3     address owner = _msgSender();
4     _transfer(owner, to, amount);
5     return true;
6 }
7 function _transfer(address from, address to,
8   uint256 amount) internal virtual {
9   require(from != address(0), "ERC20:
10    transfer from the zero address");
11   require(to != address(0), "ERC20:
12    transfer to the zero address");
13   _beforeTokenTransfer(from, to, amount);
14   uint256 fromBalance = _balances[from];
15   require(fromBalance >= amount, "ERC20:
16    transfer amount exceeds balance");
17   ...
18 }

```

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*)

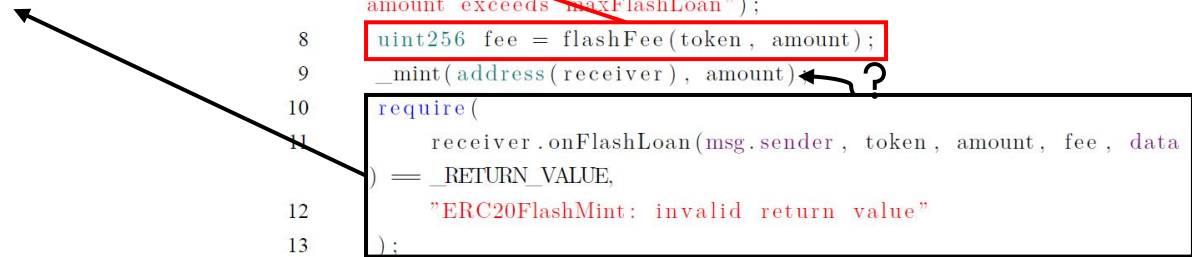
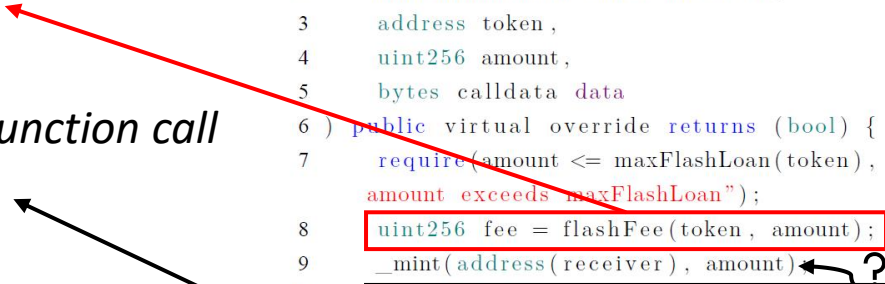
```
1 function _mint(address account, uint256 amount) internal virtual
  {
2   require(account != address(0), "ERC20: mint to the zero
   address");
3   _beforeTokenTransfer(address(0), account, amount);
4   _totalSupply += amount;
5   _balances[account] += amount;
6   emit Transfer(address(0), account, amount);
7   _afterTokenTransfer(address(0), account, amount);
8 }
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

Challenge in Mining OpenZeppelin Function Check

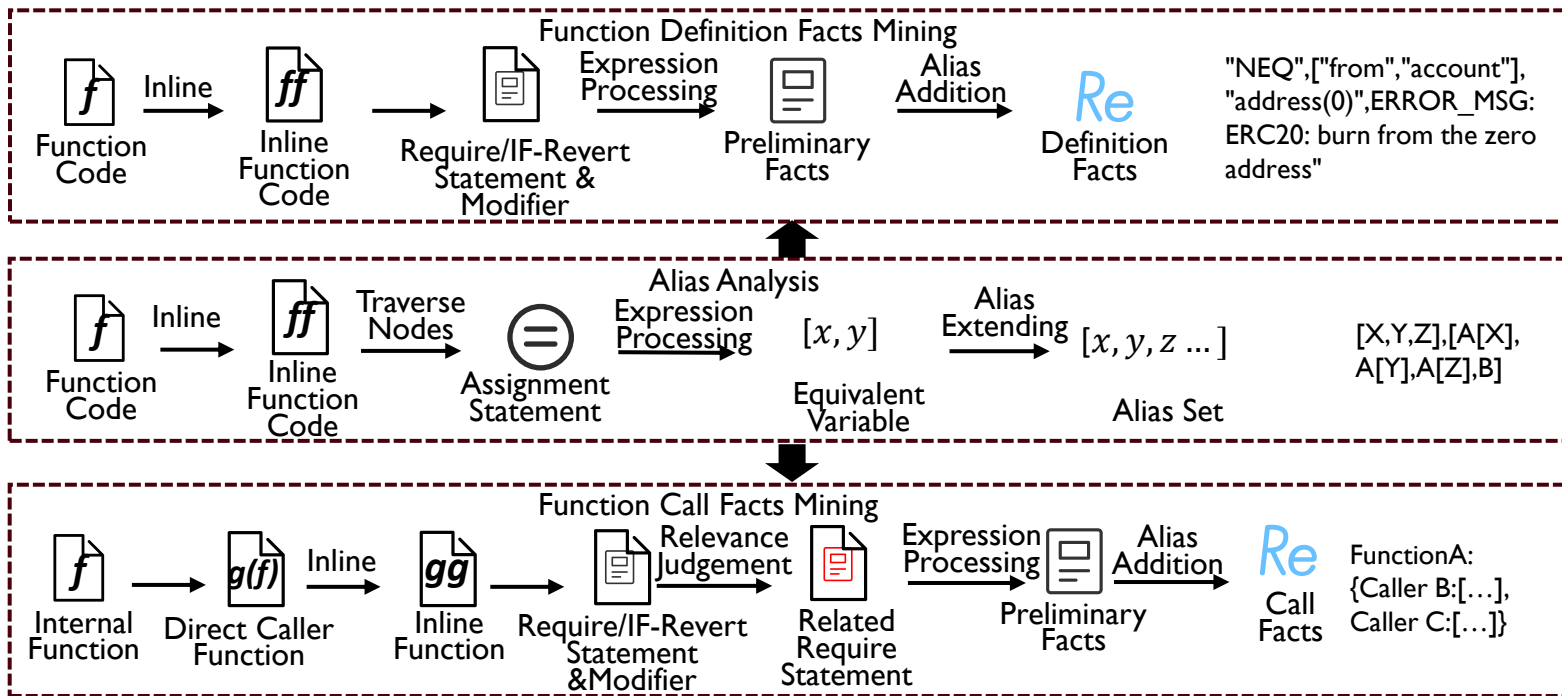
- Alias analysis needed
- Relevance between *facts after function call* and the *function call* itself.

```
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:  
8     amount exceeds maxFlashLoan");  
9     uint256 fee = flashFee(token, amount);  
10    _mint(address(receiver), amount);  
11    require(  
12        receiver.onFlashLoan(msg.sender, token, amount, fee, data  
13    ) == _RETURN_VALUE,  
14        "ERC20FlashMint: invalid return value"  
15    );  
16    address flashFeeReceiver = _flashFeeReceiver();  
17    _spendAllowance(address(receiver), address(this), amount +  
18    fee);  
19    ...  
20 }
```



3 Framework of ZepScope

Workflow of MINER



3 Framework of ZepScope

Understanding OpenZeppelin Facts

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

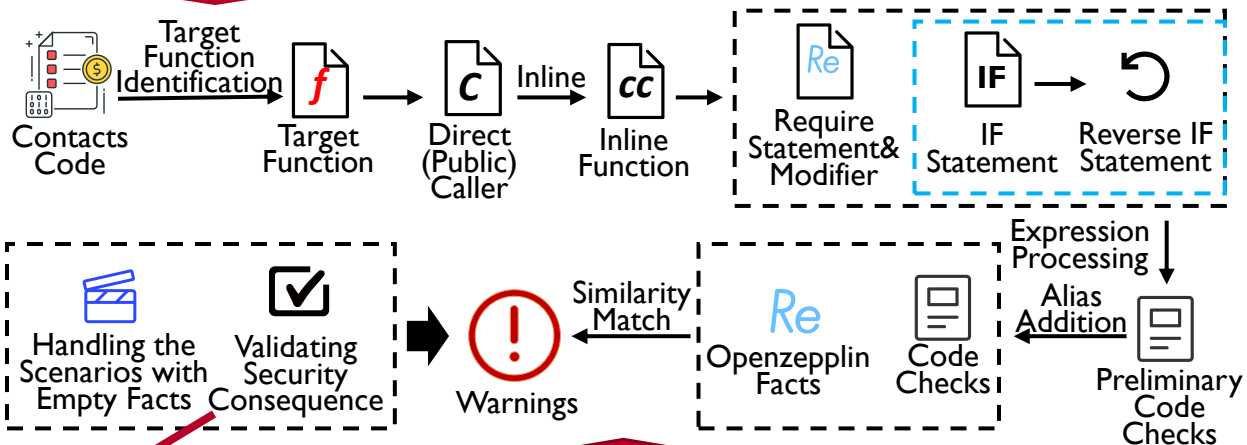
- ❑ Address Compliance Assurance
- ❑ Access Control
- ❑ Overflow/Underflow Check
- ❑ Timestamp or State Check

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

3 Framework of ZepScope

Detecting Insecure OpenZeppelin Code in SCs

Contract-Name-Included Identification & Multi-Function-Based Identification



Two Step match: Matching Error Messages & Matching the Checks

The workflow of CHECKER

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

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

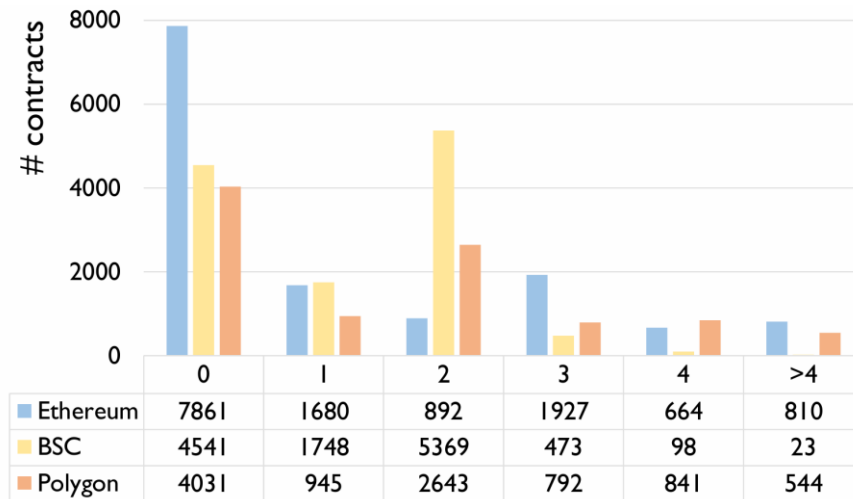
RQ2: Accuracy and Performance

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

RQ3: Security Findings

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

```
1 function transferOwnership(address newOwner)
  public virtual {
2   require(newOwner != address(0), "Ownable:
    new owner is the zero address");
3   emit OwnershipTransferred(owner, newOwner);
4   owner = newOwner;
5 }
```

```
1 function burn(uint _id) public {
2   _burn(_id);
3 }
4 function _burn(uint256 tokenId) internal
  virtual {
5   address owner = ERC721.ownerOf(tokenId);
6   _beforeTokenTransfer(owner, address(0),
    tokenId);
7   owner = ERC721.ownerOf(tokenId);
8   delete _tokenApprovals[tokenId];
9   unchecked {_balances[owner] -= 1;}
10  delete _owners[tokenId];
11  emit Transfer(owner, address(0), tokenId);
12  _afterTokenTransfer(owner, address(0),
    tokenId);
13 }
```

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

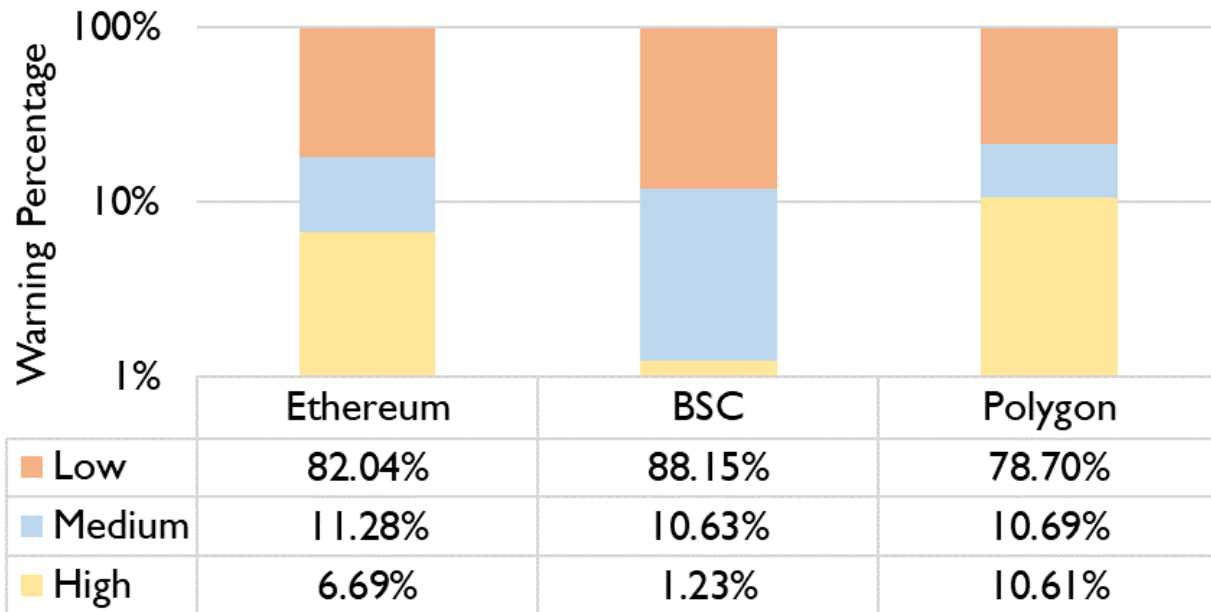
RQ3: Security Findings

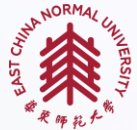
Finding 3: A Campaign of Intentionally Loosing the Checks

```
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





ECNU



香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY



NANYANG
TECHNOLOGICAL
UNIVERSITY
SINGAPORE



Thanks & QA

Han Liu¹, DaoyuanWu², Yuqiang Sun³, Haijun Wang⁴, Kaixuan Li¹, Yang Liu³, Yixiang Chen¹

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