

SAFO AUDIT



**SMART CONTRACT CODE REVIEW
SECURITY ANALYSIS REPORT FOR QBE**





TABLE OF CONTENT

| | |
|---------------------------|----|
| OVERVIEW | 1 |
| QBE PROPERTIES | 2 |
| CONTRACT FUNCTIONS | 4 |
| CONTRACT CHECKLIST | 6 |
| FUNCTIONS OF THE CONTRACT | 9 |
| GENERAL SUMMARY | 17 |
| EXECUTIVE SUMMARY | 20 |
| SEVERITY DEFINITIONS | 22 |
| AUDIT FINDINGS | 23 |
| AUTOMATED TESTING | 24 |
| AUTOMATIC GENERAL REPORT | 29 |
| AUDIT CONCLUSION | 30 |



OVERVIEW

\$QBE-\$QBE staking pool

This audit is for the below made smart contact and language of the said contract.

Project Audited

QBE Ecosystem Staking Pool QBE-QBE

Smart Contract Address

0x4E27FE7a2be6f600bb1DF7718E8a784262CA91e0

Programming Language

Solidity

Blockchain

BSC

Type

BEP-20

SAFO Audit was requested by \$QBE Ecosystem "SAFO Smart Contract Owner(s) to perform an Audit on there staking smart contract.

-The sole purpose of the audit was to achieve the following:

-Ensure that the smart contract functions for its intended purpose.

Potential security issues within the smart contract to be identified.

Use the reporting data in this report to analyze the risk exposure of the smart contract, and make improvements to its security within by addressing the identified issues.



\$QBE PROPERTIES

Smart Contract Properties

Contract Name

SafeOneChain Staking Pool \$QBE-\$QBE

Team Wallet

0x87cd02775eba9233d1b27ae5340ece05d7fdbd841

Staked Token

0xe13e2b3e521080e539260d1087c087582d1bc501

Reward Token

0xe13e2b3e521080e539260d1087c087582d1bc501

WETH

0xbb4cdb9cbd36b01bd1cbaebf2de08d9173bc095c

Update At

1681325317

Total Staked

178289055.264804456297948064

Reward Rate

2.60524562757201646

Reward Per Token Stored

0.070924398944059194

Pool Start Time

1676878253

Pool End Time

1684654253



\$QBE PROPERTIES

Smart Contract Properties

Pool Duration

7776000 (90 days)

Lock Period

259200 (3 days)

Early Unstake Fee

10%

Owner Address

0x1cc7eb8450cf70a8c03b09f9601d14ecdc06a0cf

Deployer Address

0x1cc7Eb8450cf70A8c03B09F9601d14ecdc06a0CF

Smart Contract

0x4E27FE7a2be6f600bb1DF7718E8a784262CA91e0

Blockchain

Binance Smart Chain ~ Mainnet



CONTRACT FUNCTION

Executables

1. function claimRewards() public updateReward(msg.sender)
2. function emergencyUnstake() external updateReward(msg.sender)
3. function recoverBeans() external onlyOwner
4. function recoverWrongToken(IERC20 _token) external onlyOwner
5. function setPoolDuration(uint _duration) external onlyOwner
6. function setPoolRewards(uint _amount) external onlyOwner updateReward(address(0))
7. function stake(uint _amount) external updateReward(msg.sender)
8. function topUpPoolRewards(uint _amount) external onlyOwner updateReward(address(0))
9. function transferOwnership(address _newOwner) external onlyOwner
10. function unstake(uint _amount) external updateReward(msg.sender)
11. function unwrapBeans() external onlyOwner



CONTRACT FUNCTION

Executables

```
12. function updateTeamWallet(address payable _teamWallet) external onlyOwner
```

```
13. function withdrawPoolRewards(uint256 _amount) external onlyOwner updateReward(address(0))
```

```
14. function wrapBeans() external onlyOwner
```



CONTRACT CHECKLIST

| Check | Status |
|----------------------------------|--------|
| Compiler errors | PASSED |
| Possible delays in data delivery | PASSED |
| Timestamp dependence | PASSED |
| Integer Overflow and Underflow | PASSED |
| Race Conditions and Reentrancy | PASSED |
| DoS with Revert | PASSED |
| DoS with block gas limit | PASSED |
| Methods execution permissions | PASSED |
| Economy model of the contract | PASSED |
| Private user data leaks | PASSED |



CONTRACT CHECKLIST

| Check | Status |
|---|--------|
| Malicious Events Log | PASSED |
| Scoping and Declarations | PASSED |
| Uninitialized Storage Pointers | PASSED |
| Arithmetic accuracy | PASSED |
| Design Logic | PASSED |
| Impact of the exchange rate | PASSED |
| Oracle Calls | PASSED |
| Cross-function race conditions | PASSED |
| Fallback function security | PASSED |
| Safe Open Zeppelin contracts and implementation usage | PASSED |



CONTRACT CHECKLIST

| Check | Status |
|----------------------|-----------|
| Contract correlation | UNCHECKED |
| Whitepaper | UNCHECKED |
| Website | UNCHECKED |
| Front Running. | UNCHECKED |
| Basic dApp checks | UNCHECKED |



FUNCTIONS OF THE CONTRACT

SAFE ONE CHAIN STAKING POOL STAKING CONTRACT

1. Rewards can be claimed by the user from the contract
2. The rewards have to be greater than 0 (zero)
3. If reward is \$WBNB then \$WBNB will be rewarded, otherwise the reward token is rewarded

```
function claimRewards() public updateReward(msg.sender) {
    uint rewards = userRewards[msg.sender];
    if (rewards > 0) {
        userRewards[msg.sender] = 0;
        userPaidRewards[msg.sender] += rewards;
        if(rewardToken == IERC20(0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c)) {
            WBNB.withdraw(rewards);
            (bool success, ) = msg.sender.call{value: rewards}("");
            require(success, "Transfer of beans failed.");
        } else
            rewardToken.safeTransfer(msg.sender, rewards);
        emit RewardPaid(msg.sender, rewards);
    }
}
```

4. Users can emergency unstake at any point of time
5. The users staked balance must be greater than 0 (zero)
6. An early unstake fee will be charged, will be sent over to the team wallet and the left over amount will be transferred to the executer of this function.



FUNCTIONS OF THE CONTRACT

SAFE ONE CHAIN STAKING POOL STAKING CONTRACT

```
function emergencyUnstake() external updateReward(msg.sender) {
    uint amount = userStakedBalance[msg.sender];
    if(amount <= 0) revert InvalidAmount();
    userStakedBalance[msg.sender] = 0;
    _totalStaked -= amount;
    uint fee = amount * EARLY_UNSTAKE_FEE / 10000;
    uint amountDue = amount - fee;
    stakedToken.safeTransfer(teamWallet, fee);
    stakedToken.safeTransfer(msg.sender, amountDue);
    emit Unstaked(msg.sender, amount);
}
```

7. The owner of this contact can withdraw the contract's \$BNB to their own address

```
function recoverBeans() external onlyOwner {
    uint balance = address(this).balance;
    (bool success, ) = msg.sender.call{value: balance}("");
    require(success, "Transfer failed.");
}
```

8. The owner of this contract withdraw any bep20 token from contract's balance to own address



FUNCTIONS OF THE CONTRACT

SAFE ONE CHAIN STAKING POOL STAKING CONTRACT

```
function recoverWrongToken(IERC20 _token) external onlyOwner {  
    uint balance = _token.balanceOf(address(this));  
    _token.safeTransfer(msg.sender, balance);  
}
```

9. The owner of this contract is able to change the pool duration

```
function setPoolDuration(uint _duration) external onlyOwner {  
    require(poolEndTime < block.timestamp, "Pool still live");  
    poolDuration = _duration;  
}
```



FUNCTIONS OF THE CONTRACT

SAFE ONE CHAIN STAKING POOL STAKING CONTRACT

10. The owner of this contract is able to update the “reward rate”, “pool start time”, “pool end time” and “update at time” values

```
function setPoolRewards(uint _amount) external onlyOwner updateReward(address(0)) {  
    if (_amount <= 0) revert InvalidAmount();  
    if (block.timestamp >= poolEndTime) {  
        rewardRate = _amount / poolDuration;  
    } else {  
        uint remainingRewards = (poolEndTime - block.timestamp) * rewardRate;  
        rewardRate = (_amount + remainingRewards) / poolDuration;  
    }  
    if(rewardRate <= 0) revert InvalidAmount();  
    //if(rewardRate * poolDuration > totalRewardTokens()) revert InvalidAmount();  
    poolStartTime = block.timestamp;  
    poolEndTime = block.timestamp + poolDuration;  
    updatedAt = block.timestamp;  
}
```

11. The user can stake into the pool Staking amount must be greater than zero.

12. The staking amount must be greater than 0 (zero.)



FUNCTIONS OF THE CONTRACT

SAFE ONE CHAIN STAKING POOL STAKING CONTRACT

```
function stake(uint _amount) external updateReward(msg.sender) {
    if(_amount <= 0) revert InvalidAmount();
    _totalStaked += _amount;
    userStakedBalance[msg.sender] += _amount;
    userUnlockedTime[msg.sender] = block.timestamp + LOCK_PERIOD;
    stakedToken.safeTransferFrom(msg.sender, address(this), _amount);
    emit Staked(msg.sender, _amount);
}
```

13. The owner of this contract can update the "reward rate" and "update at time"

```
function topUpPoolRewards(uint _amount) external onlyOwner updateReward(address(0)) {
    uint remainingRewards = (poolEndTime - block.timestamp) * rewardRate;
    rewardRate = (_amount + remainingRewards) / poolDuration;
    if(rewardRate <= 0) revert InvalidAmount();
    // if(stakedToken == rewardToken) {
    //     if(rewardRate * poolDuration > rewardToken.balanceOf(address(this)) - _totalStaked)
    revert InvalidAmount();
    // } else {
    //     if(rewardRate * poolDuration > rewardToken.balanceOf(address(this))) revert
    InvalidAmount();
    // }
    updatedAt = block.timestamp;
}
```



FUNCTIONS OF THE CONTRACT

SAFE ONE CHAIN STAKING POOL STAKING CONTRACT

14. The owner of this contract transfers the ownership of this contract to a “_newOwner” address

```
function transferOwnership(address _newOwner) external onlyOwner {  
    if(_newOwner == address(0)) revert InvalidAddress();  
    owner = _newOwner;  
}
```

15. Users can unstake the amount they have staked

16. The users unstake amount must be greater than zero 0 (zero)

17. The unstake time must be reached before actually unstaking

```
function unstake(uint _amount) external updateReward(msg.sender) {  
    if(block.timestamp < userUnlockedTime[msg.sender]) revert TokensLocked();  
    if(_amount <= 0) revert InvalidAmount();  
    if(_amount > userStakedBalance[msg.sender]) revert InvalidAmount();  
    _totalStaked -= _amount;  
    userStakedBalance[msg.sender] -= _amount;  
    stakedToken.safeTransfer(msg.sender, _amount);  
    emit Unstaked(msg.sender, _amount);  
}
```



FUNCTIONS OF THE CONTRACT

SAFE ONE CHAIN STAKING POOL STAKING CONTRACT

18. The owner of this contract can wrap the contract's \$BNB balance

```
function unwrapBeans() external onlyOwner {  
    uint balance = WBNB.balanceOf(address(this));  
    WBNB.withdraw(balance);  
    assert(address(this).balance >= balance);  
}
```

19. The owner of this contract can update the team wallet address

```
function updateTeamWallet(address payable _teamWallet) external onlyOwner {  
    teamWallet = _teamWallet;  
}
```



FUNCTIONS OF THE CONTRACT

SAFE ONE CHAIN STAKING POOL STAKING CONTRACT

20. The owner of this contract can withdraw the reward tokens from the pool rewards

```
function withdrawPoolRewards(uint256 _amount) external onlyOwner updateReward(address(0)) {  
    uint remainingRewards = (poolEndTime - block.timestamp) * rewardRate;  
    rewardRate = (remainingRewards - _amount) / poolDuration;  
    require(rewardRate > 0, "reward rate = 0");  
    // if(stakedToken == rewardToken) {  
    //     if(rewardRate * poolDuration > rewardToken.balanceOf(address(this)) - _totalStaked)  
revert InvalidAmount();  
    // } else {  
    //     if(rewardRate * poolDuration > rewardToken.balanceOf(address(this))) revert  
InvalidAmount();  
    // }  
    rewardToken.safeTransfer(address(msg.sender), _amount);  
    updatedAt = block.timestamp;  
}
```

21. The owner of this contract can unwrap the contract's bnb

```
function wrapBeans() external onlyOwner {  
    uint balance = address(this).balance;  
    IWETH(WBNB).deposit{value: balance}();  
    assert(IWETH(WBNB).transfer(address(this), balance));  
}
```



GENERAL SUMMARY

| Category | Subcategory | Result |
|----------------------|--|--------|
| Contract Programming | Solidity version has not specified | PASSED |
| | Solidity version is very aged | PASSED |
| | Integer overflow ~ underflow | PASSED |
| | Function input parameters lack checks | PASSED |
| | Function input parameters check bypass | PASSED |
| | Function access control lacking management | PASSED |
| | Critical operations lacking event logs | PASSED |
| | Human ~ contract checks bypass | PASSED |
| | Random number generation ~ use vulnerability | PASSED |
| | Fallback function misuse | PASSED |



GENERAL SUMMARY

| Category | Subcategory | Result |
|--------------------|--|--------|
| Code Specification | Race condition | PASSED |
| | Logical vulnerability | PASSED |
| | Other programming issues | PASSED |
| | Visibility not explicitly declared | PASSED |
| | Location not explicitly declared in Var. storage | PASSED |
| | Using keywords ~ functions to be deprecated | PASSED |
| | Other code specification issues | PASSED |
| | Assert () misuse | PASSED |
| Gas Optimization | High consumption 'for ~ while' loop | PASSED |
| | High consumption 'storage' storage | PASSED |



GENERAL SUMMARY

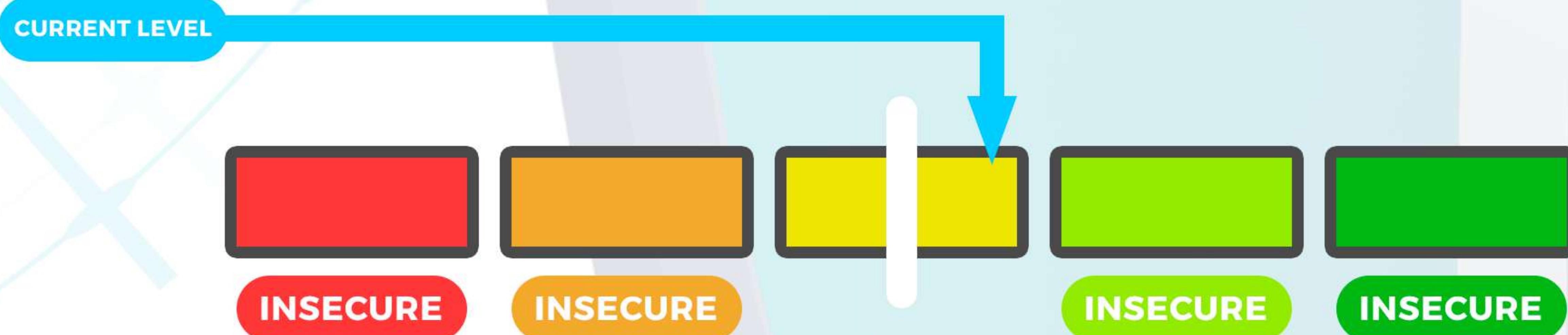
| Category | Subcategory | Result |
|----------------------|---|--------------------------------------|
| Business Risk | "Out of Gas" Attack The maximum limit for mintage not set "Short Address" Attack "Double Spend" Attack | PASSED PASSED PASSED PASSED |



EXECUTIVE SUMMARY

According to the standard audit assessment, the QBE solidity smart contract has some risks as highlighted further below.

SAFO is recommending to conclude another extensive audit assessment with a reputable third-party which will provide an additional assured conclusion.



To conduct our analysis, we utilized a range of tools such as Remix IDE and Slither amongst other instruments.

However, it's important to note that our findings are based on a thorough manual audit.

We manually reviewed all issues discovered during the automated analysis and presented any relevant vulnerabilities in the General Summary section.

| STATUS | CRITICAL | CRITICAL | CRITICAL | CRITICAL | CRITICAL |
|--------------|----------|----------|----------|----------|----------|
| OPEN | III | III | III | III | III |
| ACKNOWLEDGED | 0 | 0 | 0 | 0 | 0 |
| RESOLVED | 0 | 0 | 0 | 0 | 0 |



Code Quality

The Safe One Chain Staking Pool Smart Contract protocol consists of only one smart contract.

The libraries implemented in the Safe One Chain Staking Pool Smart Contract are part of its logical algorithm.

They are smart contracts which contain reusable code and once deployed on the blockchain (immutable ~ once only), it is assigned a specific address and its properties and/or methods can be reused many times by other contracts within the protocol.

The SAFO team has not provided scenario and unit test scripts, which would help to determine the integrity of the code in an automated way.

Overall, the code is not commented, where commenting code may provide much required documentation for functions, return variables and a lot more.

Documentation

As previously noted, incorporating comments in smart contract code is highly recommended, as they can help readers quickly comprehend the programming flow and complex code logic.

We had received a BSC explorer URL containing the Safe One Chain Staking Pool Smart Contract smart contract code.

Use of Dependencies

Based on SAFO observations, this smart contract infrastructure employs libraries that are based on well-known industry standard open-source projects.

Additionally, the core code blocks are systematically and well-written. Furthermore, this smart contract does not interact with any external smart contracts.



SEVERITY DEFINITIONS

| RISK LEVEL | DESCRIPTION |
|---|---|
| Critical | Exploiting critical vulnerabilities is often an easy task that can result in token loss, amongst other consequences. |
| High | Whilst high-level vulnerabilities may be challenging to exploit, they can have a significant impact on smart contract execution, especially if they grant public access to critical functions. |
| Update At | Whilst it's important to address medium-level vulnerabilities, it's crucial to note that they cannot result in the loss of tokens. |
| Low | Minor vulnerabilities typically stem from unused or outdated code snippets that are unlikely to have a significant impact on execution. |
| -Unknown -Lowest-level vulnerabilities -Code style violations -Style and info statements can't affect smart contracts critically | <p>It's important to note that style and best info statements won't impact smart contracts or their functionality.</p> <p>These are merely practice executions and can be safely disregarded.</p> |



AUDIT FINDINGS

Critical

There were no Critical severity vulnerabilities detected.

High

3) The owner has the ability to modify various states of the contract.

One way to make it secure is to implement the governance which will vote to make any change.

Medium

1) Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.

That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

2) Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.

Low

There were no Low severity vulnerabilities detected



AUTOMATED TESTING

SOLIDITY STATIC ANALYSIS

SOLIDITY STATIC ANALYSIS ✓ >

Select all Autorun Run

▼ Security

Select Security

- Transaction origin:
'tx.origin' used
- Check-effects-interaction:
Potential reentrancy bugs
- Inline assembly:
Inline assembly used
- Block timestamp:
Can be influenced by miners
- Low level calls:
Should only be used by experienced devs
- Block hash:
Can be influenced by miners
- Selfdestruct:
Contracts using destructed contract can
be broken

▼ Gas & Economy

Select Gas & Economy

- Gas costs:
Too high gas requirement of functions
- This on local calls:
Invocation of local functions via 'this'
- Delete dynamic array:
Use require/assert to ensure complete
deletion
- For loop over dynamic array:
Iterations depend on dynamic array's size
- Ether transfer in loop:
Transferring Ether in a for/while/do-while
loop



AUTOMATED TESTING

SOLIDITY STATIC ANALYSIS ~ CONTINUED

ERC

Select ERC

ERC20:

'decimals' should be 'uint8'

Miscellaneous

Select Miscellaneous

Constant/View/Pure functions:

Potentially constant/view/pure functions

Similar variable names:

Variable names are too similar

No return:

Function with 'returns' not returning

Guard conditions:

Ensure appropriate use of require/assert

Result not used:

The result of an operation not used

String length:

Bytes length != String length

Delete from dynamic array:

'delete' leaves a gap in array

Data truncated:

Division on int/uint values truncates the result

[IERC20Permit](#)[IERC20](#)[SafeERC20](#)[Address](#)[IWETH](#)[SafeOneChainStakingPool](#)

SOLIDITY UNIT TESTING

Test directory:

tests Create

Generate How to use...

▶ Run Stop

Select all

tests/sefo_test.sol

Progress: 1 finished (of 1)

PASS testSuite (tests/sefo_test.sol)

✓ Before all

✓ Check success

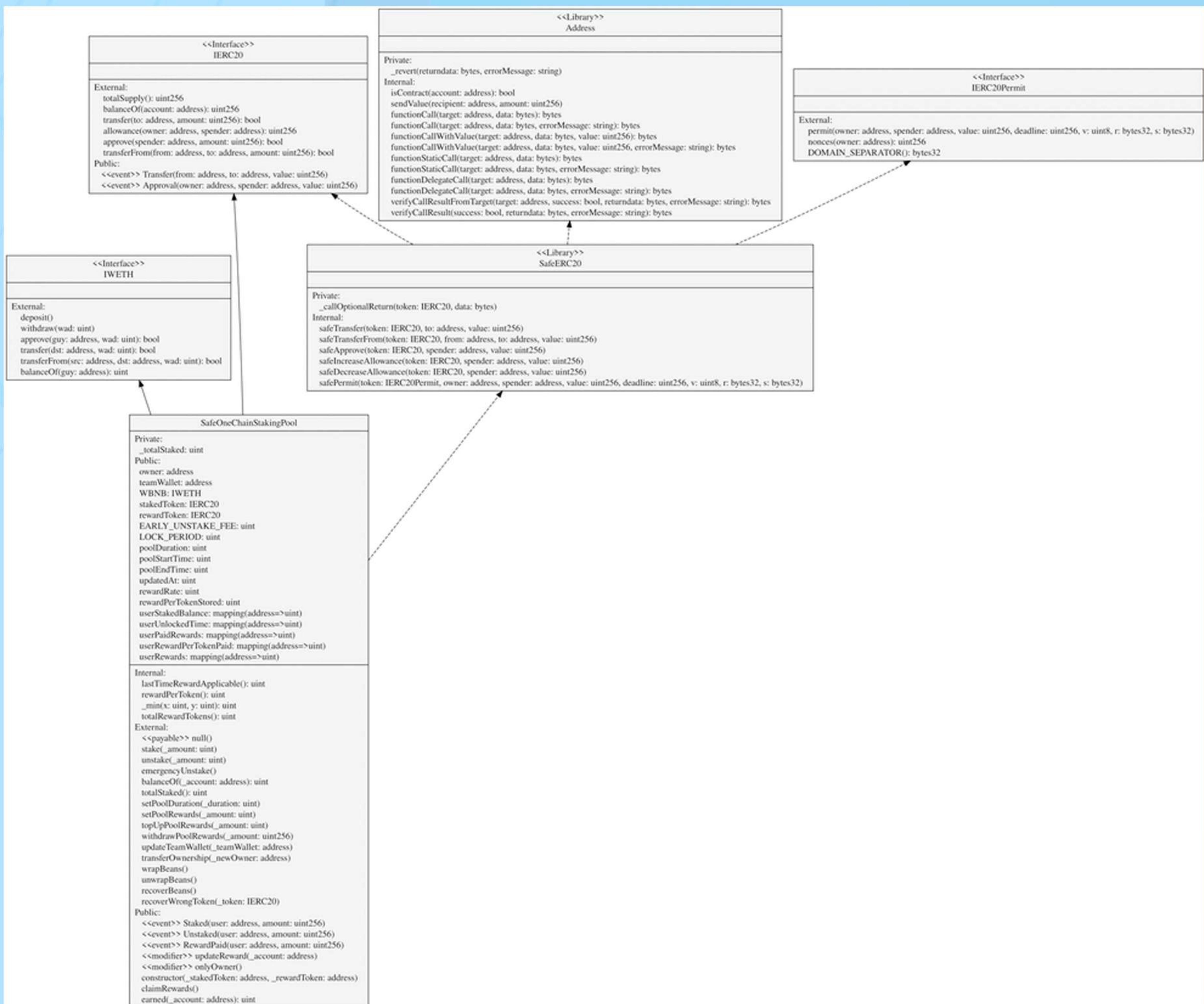
✓ Check success2

✓ Check sender and value

Result for tests/sefo_test.sol
Passed: 4
Failed: 0
Time Taken: 0.12s



UNIFIED MODELING LANGUAGE (UML)



FUNCTIONS SIGNATURE

| | | |
|----------|----|------------------------------|
| 59927044 | => | teamWallet() |
| 39efdf6b | => | EARLY_UNSTAKE_FEE() |
| 1820cabb | => | LOCK_PERIOD() |
| 8dd95002 | => | WBNB() |
| 70a08231 | => | balanceOf(address) |
| 372500ab | => | claimRewards() |
| 008cc262 | => | earned(address) |
| 7589cf2f | => | emergencyUnstake() |
| 8da5cb5b | => | owner() |
| 35c530dd | => | poolDuration() |
| 6e271dd5 | => | poolEndTime() |
| 5f96dc11 | => | poolStartTime() |
| ad08aa8f | => | recoverBeans() |
| 018bcf5c | => | recoverWrongToken(address) |
| df136d65 | => | rewardPerTokenStored() |
| 7b0a47ee | => | rewardRate() |
| f7c618c1 | => | rewardToken() |
| 02a01dc2 | => | setPoolDuration(uint256) |
| 4470c0bc | => | setPoolRewards(uint256) |
| a694fc3a | => | stake(uint256) |
| cc7a262e | => | stakedToken() |
| a9e38d36 | => | topUpPoolRewards(uint256) |
| 817b1cd2 | => | totalStaked() |
| f2fde38b | => | transferOwnership(address) |
| 2e17de78 | => | unstake(uint256) |
| e9ca7025 | => | unwrapBeans() |
| 7cb332bb | => | updateTeamWallet(address) |
| 7519ab50 | => | updatedAt() |
| c69ec430 | => | userPaidRewards(address) |
| 33e630f8 | => | userStakedBalance(address) |
| 0de01163 | => | userUnlockedTime(address) |
| ea6160ad | => | withdrawPoolRewards(uint256) |
| 3624e890 | => | wrapBeans() |



AUTOMATIC GENERAL REPORT

```

1- Files Description Table
2-
3- | File Name | SHA-1 Hash |
4- | Safe One Chain Staking Pool.sol|-----|
5- Contracts Description Table
6- | Contract | Type | Bases | |
7- | :-----:|:-----:|:-----:|:-----:|
8- | L | **Function Name** | **Visibility** | **Mutability** |
9- **Modifiers** |
10- |||||
11- | **IERC20** | Interface | |
12- | L | totalSupply | External ! | | NO ! | |
13- | L | balanceOf | External ! | | NO ! | |
14- | L | transfer | External ! | | ● NO ! | |
15- | L | allowance | External ! | | NO ! | |
16- | L | approve | External ! | | ● NO ! | |
17- | L | transferFrom | External ! | | ● NO ! | |
18- |||||
19- | **IWETH** | Interface | |
20- | L | deposit| External ! | | ● | NO ! | |
21- | L | transfer| External ! | | ● | NO ! | |
22- | L | withdraw| External ! | | ● | NO ! | |
23- | L | transferFrom| External ! | | ● NO ! | |
24- | L | approve| External ! | | ● NO ! | |
25- | L | balanceOf | External ! | | NO ! | |
26- |||||
27- | ** IERC20Permit** | Interface | |
28- | L | permit| External ! | | ● NO ! | |
29- | L | nonces | External ! | | NO ! | |
30- | L | DOMAIN_SEPARATOR| External ! | | NO ! | |
31- |||||
32- | ** SafeOneChainStakingPool** | Implementation | Ownable ||
33- | L | <Constructor> | ----- | | ● | NO ! | |
34- | L | <Receive Ether> | external ! | | ● | NO ! | |
35- | L | stake | external ! | | ● NO ! | |
36- | L | unstake | External ! | | ● NO ! | |
37- | L | emergencyUnstake | External ! | | ● NO ! | |
38- | L | claimRewards| External ! | | ● NO ! | |
Smart Contract Code Review and Security Analysis Report for
Safe One Chain Staking Pool Smart Contract
39- | L | setPoolDuration| External ! | | ● | onlyOwner |
40- | L | setPoolRewards| External ! | | ● | onlyOwner |
41- | L | topUpPoolRewards| External ! | | ● | onlyOwner |
42- | L | withdrawPoolRewards| External ! | | ● | onlyOwner |
43- | L | updateTeamWallet| External ! | | ● | onlyOwner |
44- | L | transferOwnership| External ! | | ● | onlyOwner |
45- | L | wrapBeans| External ! | | ● | onlyOwner |
46- | L | unwrapBeans| External ! | | ● | onlyOwner |
47- | L | recoverBeans| External ! | | ● | onlyOwner |
48- | L | recoverWrongToken| External ! | | ● | onlyOwner |
49-
50- Legend
51- | Symbol | Meaning |
52- | :-----:|-----|
53- | ● | Function can modify state |
54- | ● | Function is payable

```



AUDIT CONCLUSION

The Smart Contract code passed the audit successfully with some consideration to take.

There were certain warnings raised, SAFO has used all possible tests based on given information & the contract code.

Since possible test cases can be unlimited for such extensive smart contract protocol, hence we provide no such guarantee of future outcomes or the actual safety of the smart contract.

We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything. Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

Smart Contract's high-level description of functionality was presented in the General Summary section of the report.

The audit report contains all found security vulnerabilities and other issues within the reviewed code.

SAFO METHODOLOGY

SAFO prefers to work with a transparent process and make our reviews a collaborative effort.

The goals of the security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users.

The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, SAFO look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators.

SAFO also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.



Vulnerability Analysis:

SAFO audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing.

SAFO look at the project's web site to get a high-level understanding of what functionality the software under review provides.

SAFO then meet with the developers to gain an appreciation of their vision of the software.

SAFO install and use the relevant software, exploring the user interactions and roles.

While SAFO does this, we brainstorm threat models and attack surfaces.

SAFO read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

SAFO follow a conservative transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation.

Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue.

This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities.

SAFO generally, follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests.

Code analysis is the most tentative, and SAFO strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this SAFO analyze the feasibility of an attack in a live system.

Suggested Solutions:

SAFO search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases.

The mitigation and remediation:

recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after SAFO deliver our report, and before the details are made public.

