auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

SimpleStaking

DATED : 8 September 23'

MANUAL TESTING

Centralization – Unbounded lock time

Severity: High

function: openTrading

Status: Open

Overview:

Owner is able to set timePeriod (lock time) to any arbitrary value. Setting timePeriod to a large number means that stakers won't be able to unstake their tokens

```
  function setTimestamp(uint256
_timePeriodInSeconds) external onlyOwner {
      timePeriod = _timePeriodInSeconds;
  }
```

Suggestion

Set an upper limit for maximum amount of timePeriod

Example:

```
  function setTimestamp(uint256
_timePeriodInSeconds) external onlyOwner {
      require(_timePeriodInSeconds <= 14 days, “Can’t
set time period more than 14 days”);
      timePeriod = _timePeriodInSeconds;
  }
```

AUDIT SUMMARY

Project name – SimpleStaking

Date: 8 September 2023

Scope of Audit- Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

Audit Status: **Passed With High Risk**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	1	0	0	0
Acknowledged	0	0	0	0	0
Resolved	0	0	0	0	0

USED TOOLS

Tools:

1- Manual Review:
A line by line code review has been performed by audit ace team.

2- BSC Test Network: All tests were conducted on the BSC Test network, and each test has a corresponding transaction attached to it. These tests can be found in the "Functional Tests" section of the report.

3- Slither :
The code has undergone static analysis using Slither.

Testnet version:
The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address:
https://testnet.bscscan.com/address/0x532Ccd2868df9E4f50F5C4eaE2d5358a42B5D5F4#code

Token Information

Contract Address :
0x646b80C2728aa267B9f98232b79Acc0f630244DE

Name: SimpleStaking

Network: Ethereum

Token Type: ERC20

Owner: 0x7FA05f2c10c21B0f14e47446eBE41bc2CAB6d8eD

Deployer: 0x7FA05f2c10c21B0f14e47446eBE41bc2CAB6d8eD

Token Supply: 0

Checksum:
3aa85371cb9853106409d78434d3d28f551c2fad

Testnet version:

The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address: https://testnet.bscscan.com/address/0x532Ccd2868df9E4f50F5C4eaE2d5358a42B5D5F4#code

AUDIT METHODOLOGY

The auditing process will follow a routine as special considerations by Auditace:

- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.

- Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.

- Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.

- Test coverage analysis determines whether the test cases are covering the code and how much code isexercised when we run the test cases.

- Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.

- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

VULNERABILITY CHECKLIST

- ✓ Return values of low-level calls
- ✓ Private modifier
- ✓ Multiple Sends
- ✓ Using Suicide
- ✓ Gas Limitand Loops
- ✓ Address hardcoded
- ✓ Exception Disorder
- ✓ Using inline assembly
- ✓ Divide before multiply
- ✓ Missing Zero Address Validation
- ✓ Compiler version not fixed
- ✓ **Gasless Send**
- ✓ Using block.timestamp
- ✓ Re-entrancy
- ✓ Tautology or contradiction
- ✓ Timestamp Dependence
- ✓ Revert/require functions
- ✓ Use of tx.origin
- ✓ Integer overflow/underflow
- ✓ Dangerous strict equalities
- ✓ Using SHA3
- ✓ Using throw

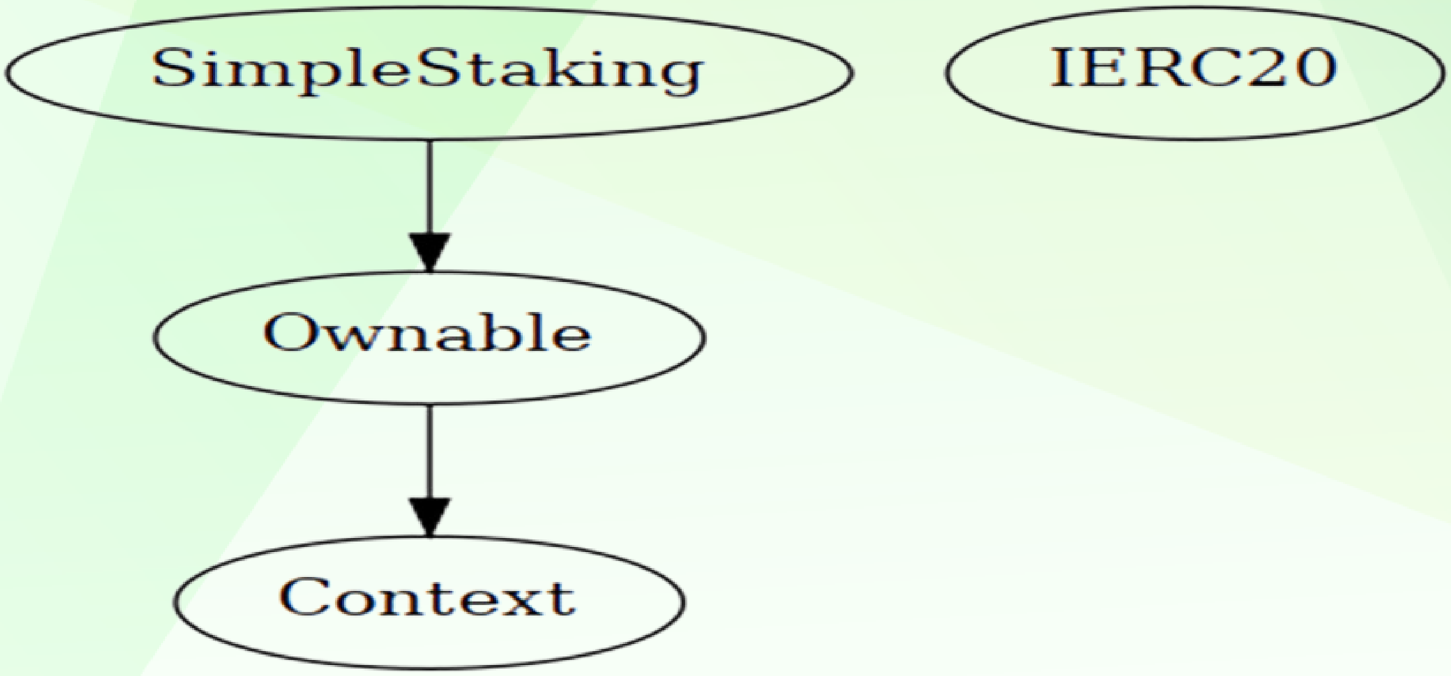
CLASSIFICATION OF RISK

Severity	Description
🟥 **Critical**	These vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
🟫 **High-Risk**	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
🟧 **Medium-Risk**	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
🟨 **Low-Risk**	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
🟦 **Gas Optimization /Suggestion**	A vulnerability that has an informational character but is not affecting any of the code.

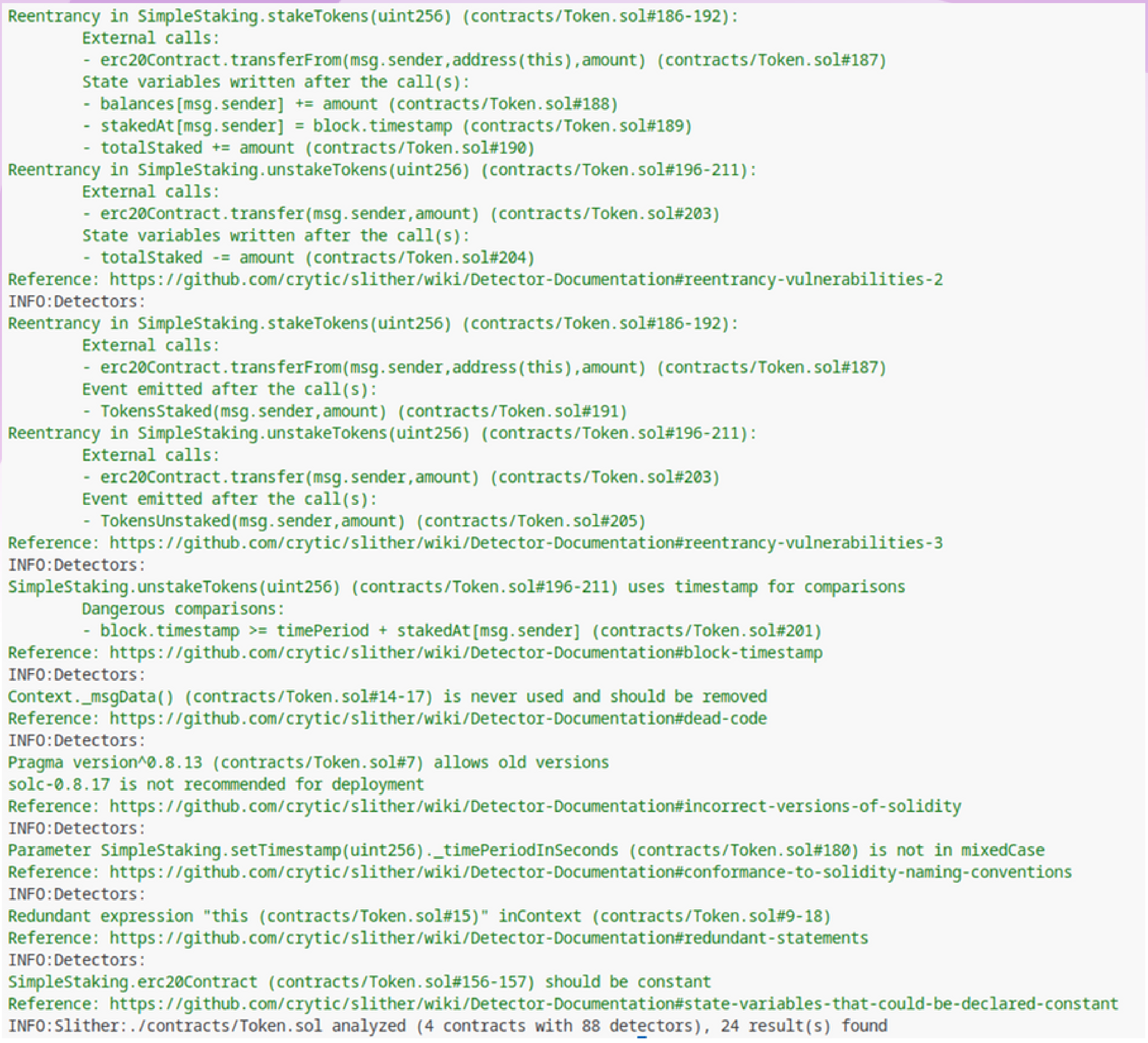
Findings

Severity	Found
🟥 **Critical**	0
🟫 **High-Risk**	1
🟧 **Medium-Risk**	0
🟨 **Low-Risk**	0
🟦 **Gas Optimization / Suggestions**	0

INHERITANCE TREE



STATIC ANALYSIS



```
Reentrancy in SimpleStaking.stakeTokens(uint256) (contracts/Token.sol#186-192):
        External calls:
        - erc20Contract.transferFrom(msg.sender,address(this),amount) (contracts/Token.sol#187)
        State variables written after the call(s):
        - balances[msg.sender] += amount (contracts/Token.sol#188)
        - stakedAt[msg.sender] = block.timestamp (contracts/Token.sol#189)
        - totalStaked += amount (contracts/Token.sol#190)
Reentrancy in SimpleStaking.unstakeTokens(uint256) (contracts/Token.sol#196-211):
        External calls:
        - erc20Contract.transfer(msg.sender,amount) (contracts/Token.sol#203)
        State variables written after the call(s):
        - totalStaked -= amount (contracts/Token.sol#204)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in SimpleStaking.stakeTokens(uint256) (contracts/Token.sol#186-192):
        External calls:
        - erc20Contract.transferFrom(msg.sender,address(this),amount) (contracts/Token.sol#187)
        Event emitted after the call(s):
        - TokensStaked(msg.sender,amount) (contracts/Token.sol#191)
Reentrancy in SimpleStaking.unstakeTokens(uint256) (contracts/Token.sol#196-211):
        External calls:
        - erc20Contract.transfer(msg.sender,amount) (contracts/Token.sol#203)
        Event emitted after the call(s):
        - TokensUnstaked(msg.sender,amount) (contracts/Token.sol#205)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
SimpleStaking.unstakeTokens(uint256) (contracts/Token.sol#196-211) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= timePeriod + stakedAt[msg.sender] (contracts/Token.sol#201)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Context._msgData() (contracts/Token.sol#14-17) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.13 (contracts/Token.sol#7) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter SimpleStaking.setTimestamp(uint256)._timePeriodInSeconds (contracts/Token.sol#180) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (contracts/Token.sol#15)" inContext (contracts/Token.sol#9-18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
SimpleStaking.erc20Contract (contracts/Token.sol#156-157) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Slither:./contracts/Token.sol analyzed (4 contracts with 88 detectors), 24 result(s) found
```

Result => A static analysis of contract’s source code has been performed using slither,

No major issues were found in the output

CONTRACT ASSESMENT

```
| Contract|      Type      |Bases    |                |                |
|:----------:|:-------------------:|:----------------:|:----------------:|:---------------:|
|    └| **Function Name** |**Visibility** | **Mutability** |**Modifiers** |
||||||
| **Context** | Implementation | |||
| └ | _msgSender | Internal 🔒 | | |
| └ | _msgData | Internal 🔒 | | |
||||||
| **IERC20** | Interface | |||
| └ | totalSupply | External ❗ | |NO ❗ |
| └ | balanceOf | External ❗ | |NO ❗ |
| └ | transfer | External ❗ | 🛑 |NO ❗ |
| └ | allowance | External ❗ | |NO ❗ |
| └ | approve | External ❗ | 🛑 |NO ❗ |
| └ | transferFrom | External ❗ | 🛑|NO ❗ |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Public ❗ | 🛑|NO ❗ |
| └ | owner | Public ❗ | |NO ❗ |
| └ | renounceOwnership | Public ❗ | 🛑| onlyOwner |
| └ | transferOwnership | Public ❗ | 🛑| onlyOwner |
||||||
| **SimpleStaking** | Implementation | Ownable |||
| └ | <Constructor> | Public ❗ | 🛑|NO ❗ |
| └ | setTimestamp | External ❗ | 🛑 | onlyOwner |
| └ | stakeTokens | External ❗ | 🛑 | noReentrant |
| └ | unstakeTokens | External ❗ | 🛑| noReentrant |
| └ | addRewards | External ❗ | 🛑|NO ❗ |
| └ | distributeRewards | External ❗ | 🛑| onlyOwner |
| └ | reduceRewards | External ❗ | 🛑| onlyOwner |
| └ | transferAccidentallyLockedTokens | External ❗ | 🛑 | onlyOwner |


### Legend

| Symbol| Meaning |
|:---------:|-----------|
|   🛑| Function can modify state |
|   💵| Function is payable |
```

FUNCTIONAL TESTING

1- Staking (passed):

https://testnet.bscscan.com/tx/0xf911100a2891b98deb9f886e25cdaa
5c53cd35e166b3f79f301a2e6c591a5699

2- Reward Distribution (passed):

https://testnet.bscscan.com/tx/0x98b37ba72eb26d74a13978fd07123
aae7495438b20df746775b6f8c78b033ccb

3- Unstaking (passed):

https://testnet.bscscan.com/tx/0x1c610909d79a1ccdea5ce22268b81d
d50f602309b110b9b309e5de3cb935fae5

MANUAL TESTING

Centralization – Unbounded lock time

Severity: High

function: openTrading

Status: Open

Overview:

Owner is able to set timePeriod (lock time) to any arbitrary value. Setting timePeriod to a large number means that stakers won't be able to unstake their tokens

```
  function setTimestamp(uint256
_timePeriodInSeconds) external onlyOwner {
      timePeriod = _timePeriodInSeconds;
  }
```

Suggestion

Set an upper limit for maximum amount of timePeriod

Example:

```
  function setTimestamp(uint256
_timePeriodInSeconds) external onlyOwner {
      require(_timePeriodInSeconds <= 14 days, “Can’t
set time period more than 14 days”);
      timePeriod = _timePeriodInSeconds;
  }
```

DISCLAIMER

ABOUT AUDITACE

We specializes in providing thorough and reliable audits for Web3 projects. With a team of experienced professionals, we use cutting-edge technology and rigorous methodologies to evaluate the security and integrity of blockchain systems. We are committed to helping our clients ensure the safety and transparency of their digital assets and transactions.

https://auditace.tech/

https://t.me/Audit_Ace

https://twitter.com/auditace_

https://github.com/Audit-Ace