CERTIK

PONZI.TRADE

SMART CONTRACT AUDIT

SMART CONTRACT SECURITY AUDIT

DISCLAIMER

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Certik and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives (Certik) owe no duty of care towards you or any other person, nor does Certik make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Certik hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Certik hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Certik, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

SMART CONTRACT STRUCTURE

IERC20

Public Functions:

totalSupply()
balanceOf(address)
transfer(address,uint256)
allowance(address,address)
approve(address,uint256)
transferFrom(address,address,uint256)

SafeMath

Private Functions:

add(uint256,uint256)
sub(uint256,uint256)
mul(uint256,uint256)
div(uint256,uint256)

TokenStaking

Public Functions:

getUserRefferalRewards(address)
invest(uint256,address)
getUserTotalDeposit(address)
getUserTotalNoDeposit(address)
getUserDepositDetail(uint256,address)
withdrawInvestment()
claimRewards()
calclulateReward(address)
getContractTokenBalance()
returnStuckTokens(address,address,uint256)
getTokenAddress()
fallback()

Private Functions:

checkExitsUser(address,address)
inspector(bool)

Public Variables:

rewardPercentage
refferalFee
adminFee
totalInvested
totalRewardWithdrwal
Stack
refferralRewards
refAddress

Private Variables:

token (IERC20)
divider

Ownable

Modifiers:

onlyOwner()

Public Variables:

_owner

Events

AUTOMATIC & MANUAL ISSUES AUDIT

The first part of the smart contract audit was done with Slither. It is a Solidity static analysis framework written in Python 3. It runs a suite of vulnerability detectors, prints visual information about contract details, and provides an API to easily write custom analyses. Slither enables developers to find vulnerabilities, enhance their code comprehension, and quickly prototype custom analyses.

Number of lines: **400**

Number of assembly lines: **0**

Number of contracts: **5**

Number of optimization issues: **0**

Number of informational issues: **17**

Number of low issues: **1**

Number of medium issues: **1**

Number of high issues: **0**

AUTOMATIC & MANUAL ISSUES AUDIT

```
----------+
|     Name     | # functions |  ERCS |     ERC20 info     | Complex code |      Feat
ures      |
+-------------+-------------+-------+--------------------+--------------+----------
----------+
|    IERC20    |      6      | ERC20 |     No Minting     |      No      |
          |
|              |             |       | Approve Race Cond. |              |
          |
|              |             |       |                    |              |                    |
|   SafeMath   |      4      |       |                    |      No      |                    |
| TokenStaking |      17     |       |                    |     Yes      | Tokens interaction |
+-------------+-------------+-------+--------------------+--------------+--------------------+
. analyzed (5 contracts)
```

High Issues: None

Medium Issues: 1

TokenStaking.calclulateReward(address) (contracts/TokenStaking.sol#304-317) performs a multiplication on the result of a division:

-reward += depositeAmount.mul(rewardPercentage).div(divider).mul(time) .div(86400) (contracts/TokenStaking.sol#310-314)

Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

AUTOMATIC & MANUAL ISSUES AUDIT

```
----------+
|     Name     | # functions |  ERCS |     ERC20 info     | Complex code |      Feat
ures      |
+-------------+-------------+-------+--------------------+--------------+----------
----------+
|    IERC20    |      6      | ERC20 |     No Minting     |      No      |
          |
|              |             |       | Approve Race Cond. |              |
          |
|              |             |       |                    |              |                    |
|   SafeMath   |      4      |       |                    |      No      |                    |
| TokenStaking |      17     |       |                    |     Yes      | Tokens interaction |
+-------------+-------------+-------+--------------------+--------------+--------------------+
. analyzed (5 contracts)
```

Low Issues: 1

TokenStaking.claimRewards() (contracts/TokenStaking.sol#254-302) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(totalRewards > 0,You dont have sufficient rewards for withdraw) (contracts/TokenStaking.sol#260-263)

Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Information Issues: 17

Pragma version0.8.17 (contracts/TokenStaking.sol#2) necessitates
a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.17 is not recommended for deployment

Reference:

https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Variable Ownable._owner (contracts/TokenStaking.sol#124) is not in mixedCase
Struct TokenStaking.deposit (contracts/TokenStaking.sol#152-156) is not in CapWords
Struct TokenStaking.stack (contracts/TokenStaking.sol#158-161) is not in CapWords
Struct TokenStaking.refRewards (contracts/TokenStaking.sol#164-167) is not in CapWords
Parameter TokenStaking.checkExitsUser(address,address)._refer
(contracts/TokenStaking.sol#177) is not in mixedCase
Parameter TokenStaking.checkExitsUser(address,address)._user
(contracts/TokenStaking.sol#177) is not in mixedCase
Parameter TokenStaking.getUserRefferalRewards(address)._address
(contracts/TokenStaking.sol#192) is not in mixedCase
Parameter TokenStaking.getUserTotalDeposit(address)._address
(contracts/TokenStaking.sol#225) is not in mixedCase
Parameter TokenStaking.getUserTotalNoDeposit(address)._address
(contracts/TokenStaking.sol#239) is not in mixedCase
Parameter TokenStaking.getUserDepositDetail(uint256,address)._address
(contracts/TokenStaking.sol#249) is not in mixedCase
Parameter TokenStaking.calclulateReward(address)._address
(contracts/TokenStaking.sol#356) is not in mixedCase
Constant TokenStaking.divider (contracts/TokenStaking.sol#142) is not in
UPPER_CASE_WITH_UNDERSCORES
Constant TokenStaking.refferalFee (contracts/TokenStaking.sol#144) is not in
UPPER_CASE_WITH_UNDERSCORES
Constant TokenStaking.adminFee (contracts/TokenStaking.sol#146) is not in
UPPER_CASE_WITH_UNDERSCORES
Variable TokenStaking.Stack (contracts/TokenStaking.sol#162) is not in mixedCase

Reference:

https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

OWNER FUNCTIONS AUDIT

The Ponzi smart contract provides people to staking their tokens per 1% percent in a day and looks secure from the smart contract owner actions. The smart contract has just 4 public (writing) functions, all of them controlled by msg.sender. As we can see, the owner doesn't have rights for smart contract interaction. The owner has one function `returnStuckTokens` which just reverts incorrect tokens from the smart contract.

THE SMART CONTRACT OWNER CAN'T

- Stop withdraw or invest functions
- Set new taxes
- Set new percentages
- Set new token address
- Withdraw main tokens from smart contact
- Withdraw BNB from smart contract
- Change owner

THE SMART CONTRACT OWNER CAN

Withdraw tokens which stuck in smart contract (not a main token)

SMART CONTRACT TESTS

The smart contract tests provide interaction with smart contracts and create some scenarios with user behavior. In our case we create 7 unit tests for interaction for 4 public functions and try to hack our system.

Token Staking smart contract

- ✓ Token Staking contract should be deployed
- ✓ GLDToken contract should be deployed
- ✓ Invest to smart contract (79ms)
- ✓ Error: Try to steal money of another person (49ms)
- ✓ Claim rewards (67ms)
- ✓ Withdraw investments (89ms)
- ✓ Error: User try to claim reward without deposit (58ms)

7 passing (4s)

CONCLUSIONS

The Ponzi smart contract is fully optimized and has minimum warnings. The architecture is very clean and the smart contract owner does not have any serious rights. The smart contract provides 1% of tokens every day and can't be stopped and hacked.

Recommendations:

1. Consider ordering multiplication before division.
2. block.timestamp. block.timestamp can be manipulated by miners.
3. Follow the Solidity naming convention.

Certik note: Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.