Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

# SMART CONTRACT

## Security Audit Report

Project: Tyrion Staking
Website: https://www.tyrion.io
Platform: Binance Smart Chain
Language: Solidity
Date: September 16th, 2023

# Table of contents




**Email: audit@EtherAuthority.io**

`

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.


**Email: audit@EtherAuthority.io**

# Introduction

EtherAuthority was contracted by the Tyrion team to perform the Security audit of the Tyrion Staking smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on September 16th, 2023.

**The purpose of this audit was to address the following:**

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

# Project Background

- Tyrion is an innovative cryptocurrency ecosystem.
- The Tyrion Staking is a smart contract that has functionalities of stake/unstake tokens, rewards in the form of tokens.

# Audit scope

| **Name** | **Code Review and Security Analysis Report for Tyrion Staking Smart Contract** |
|---|---|
| **Platform** | **BSC / Solidity** |
| **File** | Tyrion_Staking.sol |
| **Online code** | 0xFC6964d9e0141e7CFc37081A2B45429C17DCDd3d |
| **Updated MD5 hash** | 4538D947C69F4611F248A99070985770 |
| **Audit Date** | September 16th, 2023 |
| **Revised Audit Date** | September 19th, 2023 |

## Claimed Smart Contract Features

| Claimed Feature Detail | Our Observation |
| --- | --- |
| • Apy Timeframe: 1 day<br>• Lockup Period: 365 days<br>• Unstake request time: 14 days<br>• APY: 25%<br>• Minimum APY: 3%<br>• Penality: 10%<br>• Maximum investment: 1,00,000 TYON<br>• Minimum withdraw reward limit: 10 TYON<br>• Minimum investment: 100 TYON<br>• Total reward to distribute: 50 Million TYON | **YES, This is valid.**<br>**The smart contract owner controls these functions, so the owner must handle the private key of the owner's wallet very securely.**<br>**Because if the private key is compromised, then it will create problems.** |
| **Ownership Control:**<br>• Withdraw funds.<br>• Update APY values.<br>• Update investment values.<br>• Update penality.<br>• Update withdrawal limit.<br>• Update Lockup period.<br>• Update Distributed reward.<br>• Current owner can transfer the ownership. | **YES, This is valid.** |

# Audit Summary

According to the standard audit assessment, Customer`s solidity based smart contracts are **“Secured”**. Also, these contracts contain owner control, which does not make them fully decentralized.


We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

**We found 0 critical, 0 high, 1 medium and 1 low and 8 very low level issues.**
**We confirm that these all issues are fixed / acknowledged in the revised smart contract code.**

**Investors Advice:** Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

# Technical Quick Stats

| Main Category | Subcategory | Result |
|---|---|---|
| Contract Programming | Solidity version not specified | Passed |
| | Solidity version too old | Passed |
| | Integer overflow/underflow | Passed |
| | Function input parameters lack of check | Moderated |
| | Function input parameters check bypass | Passed |
| | Function access control lacks management | Passed |
| | Critical operation lacks event log | Passed |
| | Human/contract checks bypass | Passed |
| | Random number generation/use vulnerability | N/A |
| | Fallback function misuse | Passed |
| | Race condition | Passed |
| | Logical vulnerability | Passed |
| | Features claimed | Moderated |
| | Other programming issues | Passed |
| Code Specification | Function visibility not explicitly declared | Passed |
| | Var. storage location not explicitly declared | Passed |
| | Use keywords/functions to be deprecated | Passed |
| | Unused code | Moderated |
| Gas Optimization | “Out of Gas” Issue | Passed |
| | High consumption ‘for/while’ loop | Passed |
| | High consumption ‘storage’ storage | Passed |
| | Assert() misuse | Passed |
| Business Risk | The maximum limit for mintage not set | Passed |
| | “Short Address” Attack | Passed |
| | “Double Spend” Attack | Passed |

**Overall Audit Result:** **PASSED**


**Email: audit@EtherAuthority.io**

## Code Quality

This audit scope has 1 smart contract. Smart contract contains Libraries, Smart contracts, inherits and Interfaces.  This is a compact and well written smart contract.

The libraries in Tyrion Staking are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the Tyrion Staking.

The Tyrion Staking team has **not** provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are not **well** commented on in the smart contracts. Ethereum's NatSpec commenting style is recommended.

## Documentation

We were given a Tyrion Staking smart contract code in the form of a bscscan.com web link. The hash of that code is mentioned above in the table.

As mentioned above, code parts are not **well** commented. but The logic is straightforward. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official project URL: https://www.tyrion.io  which provided rich information about the project architecture.

## Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects.

Apart from libraries,  its functions are not used in external smart contract calls.

# AS-IS overview

### Functions

| Sl. | Functions | Type | Observation | Conclusion |
|---|---|---|---|---|
| 1 | constructor | write | Passed | No Issue |
| 2 | Stake | external | Ambiguous Error Message | Refer Audit Findings |
| 3 | get_apy_temp | read | Passed | No Issue |
| 4 | get_apy | read | Passed | No Issue |
| 5 | get_TotalReward | read | Passed | No Issue |
| 6 | getReward_perInv | read | Passed | No Issue |
| 7 | withdrawReward | external | Passed | No Issue |
| 8 | unStake | external | Unstake request not updated | Refer Audit Findings |
| 9 | unStake_Request | external | Unstake request not updated | Refer Audit Findings |
| 10 | get_ReqEndTime | read | Passed | No Issue |
| 11 | getTotalInvestment | read | Passed | No Issue |
| 12 | getAll_investments | read | Passed | No Issue |
| 13 | getAll_investments_ForReward | read | Passed | No Issue |
| 14 | transferOwnership | write | access only Owner | No Issue |
| 15 | total_withdraw_reaward | read | Passed | No Issue |
| 16 | get_currTime | read | Passed | No Issue |
| 17 | get_withdrawnTime | read | Passed | No Issue |
| 18 | withdrawFunds | write | access only Owner | No Issue |
| 19 | update_minimum_Apy | write | Value limit is not set | Refer Audit Findings |
| 20 | update_minimum_investment | write | Value limit is not set | Refer Audit Findings |
| 21 | update_max_investment | write | Value limit is not set | Refer Audit Findings |
| 22 | update_penality | write | Value limit is not set | Refer Audit Findings |
| 23 | update_withdraw_limit | write | Value limit is not set | Refer Audit Findings |
| 24 | update_Lockup_period | write | Value limit is not set | Refer Audit Findings |
| 25 | update_Apy_Timeframe | write | Value limit is not set | Refer Audit Findings |
| 26 | update_distributed_reward | write | Value limit is not set | Refer Audit Findings |
| 27 | update_Unstake_request_time | write | access only Owner | No Issue |
| 28 | update_APY | write | Value limit is not set | Refer Audit Findings |
| 29 | onlyOwner | modifier | Passed | No Issue |

# Severity Definitions

| Risk Level | Description |
|---|---|
| Critical | Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc. |
| High | High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial |
| Medium | Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose |
| Low | Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution |
| Lowest / Code Style / Best Practice | Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored. |


**Email: audit@EtherAuthority.io**

# Audit Findings

## Critical Severity

No Critical severity vulnerabilities were found.

## High Severity

No High severity vulnerabilities were found.

## Medium

(1) Value limit is not set:

```
function update_max_investment(uint _value)  public    24776 gas
{
    require(msg.sender==owner);

    maximum_investment = _value;
}

function update_penality(uint _value)  public    24776 gas
{
    require(msg.sender==owner);

    penality = _value;
}
```

```
function update_withdraw_limit(uint _value)  public    24799 gas
{
    require(msg.sender==owner);

    minimum_withdraw_reward_limit = _value;
}

function update_Lockup_period(uint _value)  public    24777 gas
{
    require(msg.sender==owner);

    Lockup_period = _value;
}
```

```
function update_Apy_Timeframe(uint _value)  public    24821 gas
{
    require(msg.sender==owner);

    Apy_Timeframe = _value;
}
```

```
function update_distributed_reward(uint _value)  public
{
    require(msg.sender==owner);

    total_reward_to_distribute = _value;
}
```

In the listed functions, variable values can be set with any number.

An explicit range limit should be set for each variable according to its use in calculations:

- update_minimum_Apy
- update_minimum_investment
- update_max_investment
- update_penality
- update_withdraw_limit
- update_Lockup_period
- update_Apy_Timeframe
- update_APY
- update_distributed_reward

**Resolution:** Consider adding an explicit value to the values.

**Status:** **Acknowledged**

## Low

(1) Critical operation lacks event log:

Missing event log for:

- Stake
- Unstake
- withdrawReward
- withdrawFunds

- unStake_Request

**Resolution:** Write an event log for the listed events.

**Status:** **Fixed**

## Very Low / Informational / Best practices:

(1) Hard Coded variable values:

```
address  public owner;
address Staking_token=0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8; //credit
address Reward_Token =0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8; // bel3
```

Staking_token and Reward_Token variables are set with hardcoded values.

**Resolution:** We advise always ensuring the set hardcoded values.

**Status:** **Acknowledged**

(2) Missing error message:

```
function withdrawFunds(uint _amount)  public    infinite gas
{
    require(msg.sender==owner);
    uint bal = Token(Staking_token).balanceOf(address(this));
    require(bal>=_amount);
    Token(Staking_token).transfer(owner,_amount);
}
```

Error messages are missing in some functions. Requirements must have error messages.

**Resolution:** We suggest adding appropriate error messages.

**Status:** **Fixed**

(3) Make variables constant:

Staking_token and Reward_Token: These variable values will be unchanged. So, please make it constant. It will save some gas.

**Resolution:** We advise declaring those variables as constants. Just use a constant keyword. And define constants in the constructor.

**Status:** **Fixed**

(4) Unstake request not updated:

After executing unStake_Request user, unstake request details have not been updated with user information.

**Resolution:** We suggest checking code logic and making appropriate changes to update the information.

**Status:** **Acknowledged**

(5) Ambiguous Error Message:

```
function Stake(uint _investedamount, bool _autoCompounding) external  returns(bool success)   infinite gas
{
    require(_investedamount >= minimum_investment && _investedamount <= maximum_investment ,"value is not greater than 0");

    require(Token(Staking_token).allowance(msg.sender,address(this))>=_investedamount,"allowance");


    if(user[msg.sender].investBefore == false)
    {
        All_investors[totalusers]=msg.sender;
        isUser[msg.sender]=true;
```

The mentioned error message does not explain exactly the error of the operation.

**Resolution:** As error messages are intended to notify users about failing conditions, they should provide enough information so that appropriate corrections can be made to interact with the system.

**Status:** **Acknowledged**

(6) Features claimed:

```
    uint public total_reward_to_distribute = 50000000*10**9;
```

In document, there are two values

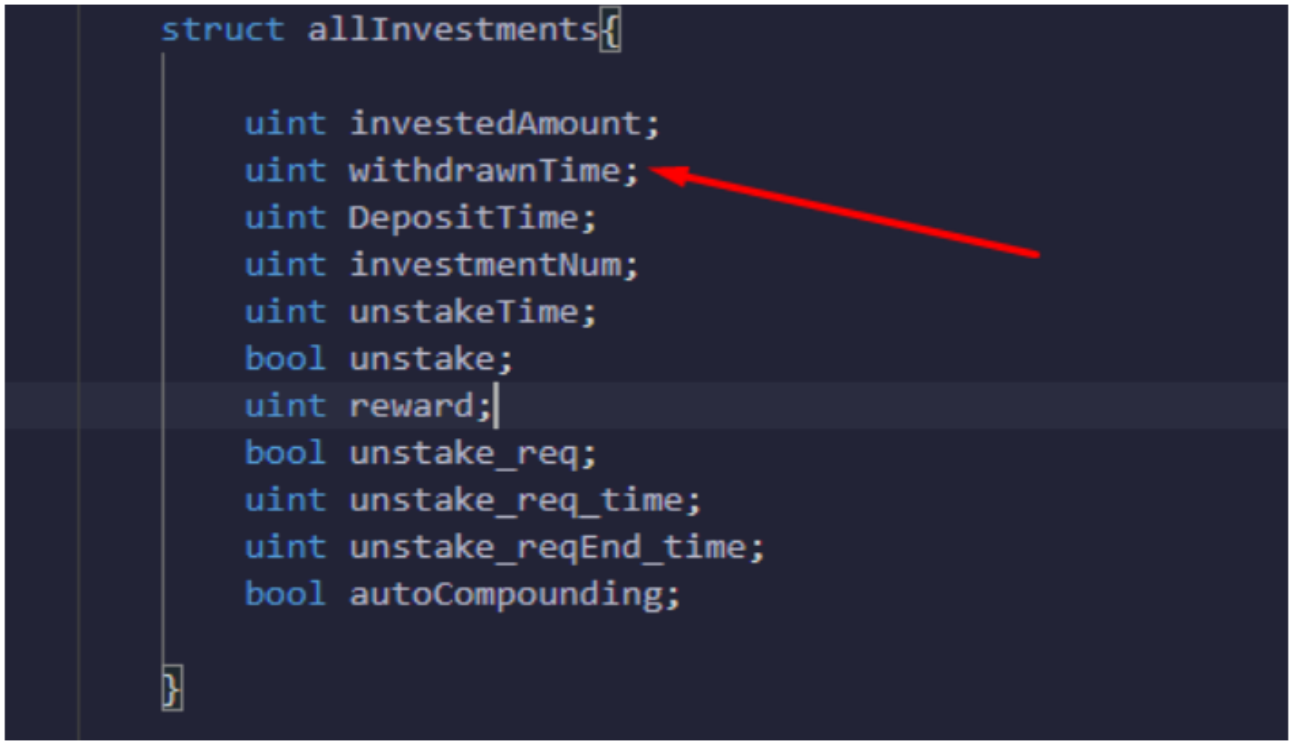
1. Maximum rewards = 3 million
2. Total supply = 500 million

But In code total_reward_to_distribute is set by 50 million.

**Resolution:** We suggest confirming for the total rewards to be distributed.

**Status:** Acknowledged

(7) Unused variable:




The withdrawnTime is allInvestments variable; its value is set in the stake function but never used anywhere in the contract.

**Resolution:** We suggest removing all unused variables.

**Status:** Fixed

(8) Add “OnlyOwner” modifier:

```
function update_minimum_Apy(uint _value)  public    24733 gas
{
    require(msg.sender==owner);

    minimum_Apy = _value;
}

function update_minimum_investment(uint _value)  public    24731 gas
{
    require(msg.sender==owner);

    minimum_investment = _value;
}
```

```
function update_max_investment(uint _value)  public    24776 gas
{
    require(msg.sender==owner);

    maximum_investment = _value;
}

function update_penality(uint _value)  public    24776 gas
{
    require(msg.sender==owner);

    penality = _value;
}

function update_withdraw_limit(uint _value)  public    24799 gas
{
    require(msg.sender==owner);

    minimum_withdraw_reward_limit = _value;
}
```

```
function update_Lockup_period(uint _value)  public    24777 gas
{
    require(msg.sender==owner);

    Lockup_period = _value;
}

function update_Apy_Timeframe(uint _value)  public    24821 gas
{
    require(msg.sender==owner);

    Apy_Timeframe = _value;
}
```

Below, All the setter functions are accessible to only the owner. To check this, the required statement is written in all the setter functions.

- transferOwnership
- withdrawFunds
- update_minimum_Apy
- update_minimum_investment
- update_max_investment
- update_penality
- update_withdraw_limit
- update_Lockup_period
- update_Apy_Timeframe
- update_distributed_reward
- update_Unstake_request_time
- update_APY

**Resolution:** We advise adding the "OnlyOwner" modifier to check this accessibility, set the modifier in all the functions, and remove the required statements.

**Status:** **Fixed**

# Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

### Tyrion_Staking.sol

- transferOwnership: The current owner can transfer ownership of the contract to a new account.
- withdrawFunds: The owner can withdraw funds.
- update_minimum_Apy: Minimum APY values can be updated by the owner.
- update_minimum_investment: Minimum investment values can be updated by the owner.
- update_max_investment: Maximum investment values can be updated by the owner.
- update_penality: Penality values can be updated by the owner.
- update_withdraw_limit: Withdraw limit values can be updated by the owner.
- update_Lockup_period: Lockup period values can be updated by the owner.
- update_Apy_Timeframe: APY Timeframe values can be updated by the owner.
- update_distributed_reward: Distributed reward values can be updated by the owner.
- update_Unstake_request_time: Unstake request time can be updated by the owner.
- update_APY: APY values can be updated by the owner.

To make the smart contract 100% decentralized, we suggest renouncing ownership of the smart contract once its function is completed.

# Conclusion

We were given a contract code in the form of a bscscan.com web link. And we have used all possible tests based on given objects as files. We had observed 1 medium, 1 low and 8 informational issues in the smart contracts.but those are not critical ones. We confirm that these all issues are fixed / acknowledged in the revised smart contract code. So, **it's good to go for the production**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. 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 As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed smart contract, based on standard audit procedure scope, is **"Secured".**

# Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our 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, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We 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:**

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We 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:**

We 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. We 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 we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

**Suggested Solutions:**

We 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 we deliver our report, and before the details are made public.

# Disclaimers

## EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

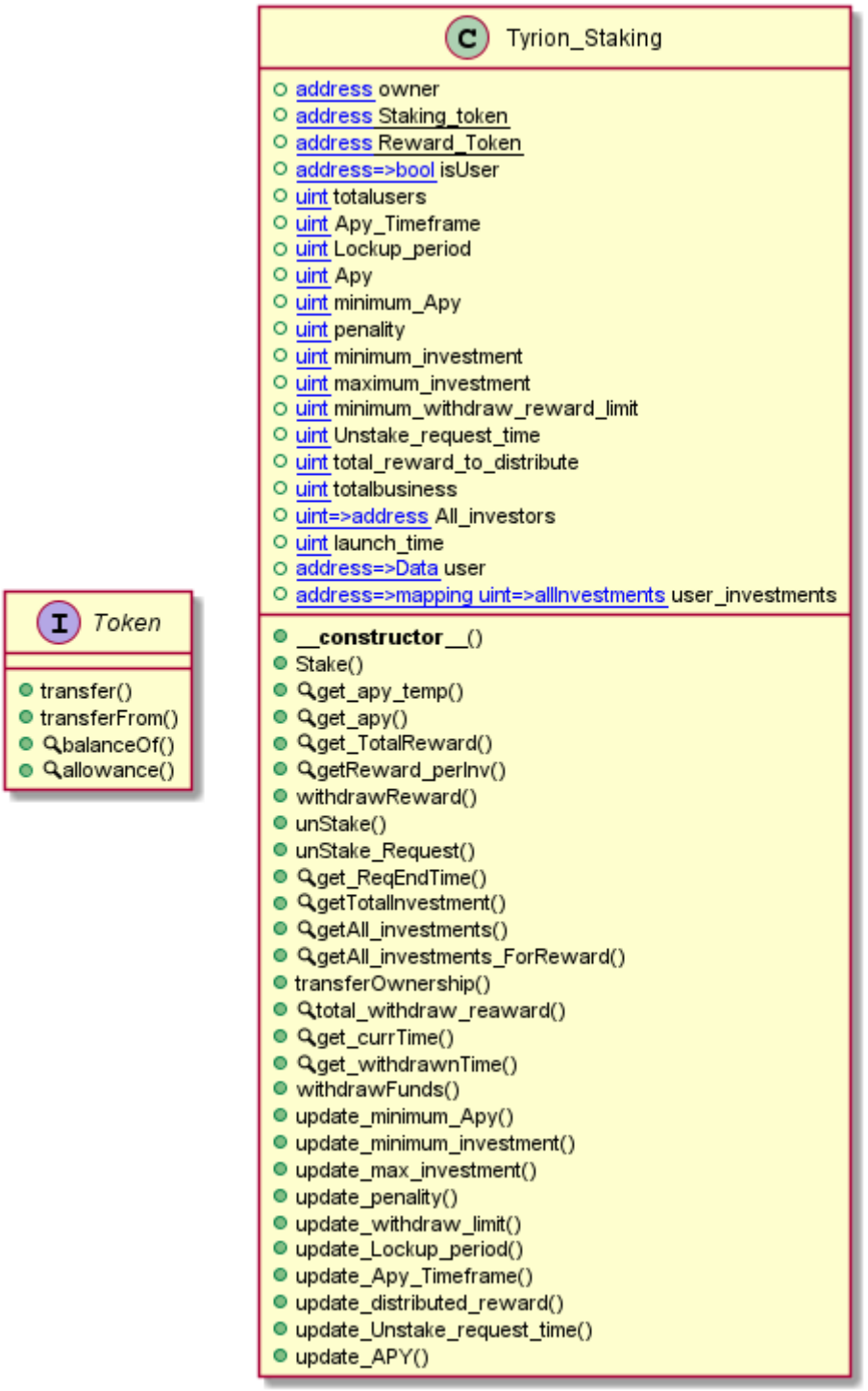
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

## Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.

# Appendix

## Code Flow Diagram - Tyrion Staking



# Slither Results Log

Slither is a Solidity static analysis framework that uses vulnerability detectors, displays contract details, and provides an API for writing custom analyses. It helps developers identify vulnerabilities, improve code comprehension, and prototype custom analyses quickly. The analysis includes a report with warnings and errors, allowing developers to quickly prototype and fix issues.

We did the analysis of the project altogether. Below are the results.

### Slither Log >> Tyrion_Staking.sol

```
Tyrion_Staking.update_minimum_Apy(uint256) (Tyrion_Staking.sol#472-477) should emit an event for:
        - minimum_Apy = _value (Tyrion_Staking.sol#476)
Tyrion_Staking.update_minimum_investment(uint256) (Tyrion_Staking.sol#479-484) should emit an event for:
        - minimum_investment = _value (Tyrion_Staking.sol#483)
Tyrion_Staking.update_max_investment(uint256) (Tyrion_Staking.sol#486-491) should emit an event for:
        - maximum_investment = _value (Tyrion_Staking.sol#490)
Tyrion_Staking.update_penality(uint256) (Tyrion_Staking.sol#493-498) should emit an event for:
        - penality = _value (Tyrion_Staking.sol#497)
Tyrion_Staking.update_withdraw_limit(uint256) (Tyrion_Staking.sol#500-505) should emit an event for:
        - minimum_withdraw_reward_limit = _value (Tyrion_Staking.sol#504)
Tyrion_Staking.update_Lockup_period(uint256) (Tyrion_Staking.sol#507-512) should emit an event for:
        - Lockup_period = _value (Tyrion_Staking.sol#511)
Tyrion_Staking.update_Apy_Timeframe(uint256) (Tyrion_Staking.sol#514-519) should emit an event for:
        - Apy_Timeframe = _value (Tyrion_Staking.sol#518)
Tyrion_Staking.update_distributed_reward(uint256) (Tyrion_Staking.sol#520-525) should emit an event for:
        - total_reward_to_distribute = _value (Tyrion_Staking.sol#524)
Tyrion_Staking.update_Unstake_request_time(uint256) (Tyrion_Staking.sol#527-532) should emit an event for:
        - Unstake_request_time = _value (Tyrion_Staking.sol#531)
Tyrion_Staking.update_APY(uint256) (Tyrion_Staking.sol#534-539) should emit an event for:
        - Apy = _value (Tyrion_Staking.sol#538)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Tyrion_Staking.transferOwnership(address)._owner (Tyrion_Staking.sol#432) lacks a zero-check on :
                - owner = _owner (Tyrion_Staking.sol#435)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Reentrancy in Tyrion_Staking.Stake(uint256,bool) (Tyrion_Staking.sol#89-125):
        External calls:
        - Token(Staking_token).transferFrom(msg.sender,address(this),_investedamount) (Tyrion_Staking.sol#119)
        State variables written after the call(s):
        - user_investments[msg.sender][num] = user[msg.sender].investment[num] (Tyrion_Staking.sol#120)
Reentrancy in Tyrion_Staking.unStake(uint256) (Tyrion_Staking.sol#327-353):
        External calls:
        - Token(Staking_token).transfer(owner,penalty_fee) (Tyrion_Staking.sol#339)
        - Token(Staking_token).transfer(msg.sender,amount) (Tyrion_Staking.sol#342)
        State variables written after the call(s):
        - user_investments[msg.sender][num] = user[msg.sender].investment[num] (Tyrion_Staking.sol#348)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Tyrion_Staking.get_TotalReward() (Tyrion_Staking.sol#208-261) uses timestamp for comparisons
        Dangerous comparisons:
        - depTime > 0 (Tyrion_Staking.sol#233)
        - j < depTime (Tyrion_Staking.sol#237)
Tyrion_Staking.getReward_perInv(uint256) (Tyrion_Staking.sol#263-312) uses timestamp for comparisons
        Dangerous comparisons:
        - depTime > 0 (Tyrion_Staking.sol#285)
        - j < depTime (Tyrion_Staking.sol#289)
Tyrion_Staking.unStake(uint256) (Tyrion_Staking.sol#327-353) uses timestamp for comparisons
        Dangerous comparisons:
        - (user[msg.sender].investment[num].unstake_req && user[msg.sender].investment[num].unstake_reqEnd_time > block.timest
amp) || ! user[msg.sender].investment[num].unstake_req && user[msg.sender].investment[num].unstake_req_time == 0 (Tyrion_Staki
ng.sol#336)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Tyrion_Staking.Stake(uint256,bool) (Tyrion_Staking.sol#89-125) compares to a boolean constant:
        -user[msg.sender].investBefore == false (Tyrion_Staking.sol#96)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```

```
Tyrion_Staking.Stake(uint256,bool) (Tyrion_Staking.sol#89-125) compares to a boolean constant:
        -user[msg.sender].investBefore == false (Tyrion_Staking.sol#96)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Pragma version^0.8.0 (Tyrion_Staking.sol#7) allows old versions
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Contract Tyrion_Staking (Tyrion_Staking.sol#16-542) is not in CapWords
Struct Tyrion_Staking.allInvestments (Tyrion_Staking.sol#42-56) is not in CapWords
Function Tyrion_Staking.Stake(uint256,bool) (Tyrion_Staking.sol#89-125) is not in mixedCase
Parameter Tyrion_Staking.Stake(uint256,bool)._investedamount (Tyrion_Staking.sol#89) is not in mixedCase
Parameter Tyrion_Staking.Stake(uint256,bool)._autoCompounding (Tyrion_Staking.sol#89) is not in mixedCase
Function Tyrion_Staking.get_apy_temp() (Tyrion_Staking.sol#127-165) is not in mixedCase
Function Tyrion_Staking.get_apy() (Tyrion_Staking.sol#166-204) is not in mixedCase
Function Tyrion_Staking.get_TotalReward() (Tyrion_Staking.sol#208-261) is not in mixedCase
Function Tyrion_Staking.getReward_perInv(uint256) (Tyrion_Staking.sol#263-312) is not in mixedCase
Function Tyrion_Staking.unStake_Request(uint256) (Tyrion_Staking.sol#357-369) is not in mixedCase
Function Tyrion_Staking.get_ReqEndTime(uint256) (Tyrion_Staking.sol#373-376) is not in mixedCase
Function Tyrion_Staking.getAll_investments() (Tyrion_Staking.sol#384-412) is not in mixedCase
Function Tyrion_Staking.getAll_investments_ForReward() (Tyrion_Staking.sol#414-429) is not in mixedCase
Parameter Tyrion_Staking.transferOwnership(address)._owner (Tyrion_Staking.sol#432) is not in mixedCase
Function Tyrion_Staking.total_withdraw_reaward() (Tyrion_Staking.sol#438-446) is not in mixedCase
Function Tyrion_Staking.get_currTime() (Tyrion_Staking.sol#447-450) is not in mixedCase
Function Tyrion_Staking.get_withdrawnTime(uint256) (Tyrion_Staking.sol#452-455) is not in mixedCase
Parameter Tyrion_Staking.withdrawFunds(uint256)._amount (Tyrion_Staking.sol#458) is not in mixedCase
Function Tyrion_Staking.update_minimum_Apy(uint256) (Tyrion_Staking.sol#472-477) is not in mixedCase
Parameter Tyrion_Staking.update_minimum_Apy(uint256)._value (Tyrion_Staking.sol#472) is not in mixedCase
Function Tyrion_Staking.update_minimum_investment(uint256) (Tyrion_Staking.sol#479-484) is not in mixedCase
Parameter Tyrion_Staking.update_minimum_investment(uint256)._value (Tyrion_Staking.sol#479) is not in mixedCase
Function Tyrion_Staking.update_max_investment(uint256) (Tyrion_Staking.sol#486-491) is not in mixedCase
Parameter Tyrion_Staking.update_max_investment(uint256)._value (Tyrion_Staking.sol#486) is not in mixedCase
Function Tyrion_Staking.update_penality(uint256) (Tyrion_Staking.sol#493-498) is not in mixedCase
Parameter Tyrion_Staking.update_penality(uint256)._value (Tyrion_Staking.sol#493) is not in mixedCase
Function Tyrion_Staking.update_withdraw_limit(uint256) (Tyrion_Staking.sol#500-505) is not in mixedCase
Parameter Tyrion_Staking.update_withdraw_limit(uint256)._value (Tyrion_Staking.sol#500) is not in mixedCase
Function Tyrion_Staking.update_Lockup_period(uint256) (Tyrion_Staking.sol#507-512) is not in mixedCase
Parameter Tyrion_Staking.update_Lockup_period(uint256)._value (Tyrion_Staking.sol#507) is not in mixedCase
Function Tyrion_Staking.update_Apy_Timeframe(uint256) (Tyrion_Staking.sol#514-519) is not in mixedCase
Parameter Tyrion_Staking.update_Apy_Timeframe(uint256)._value (Tyrion_Staking.sol#514) is not in mixedCase
Function Tyrion_Staking.update_distributed_reward(uint256) (Tyrion_Staking.sol#520-525) is not in mixedCase
Parameter Tyrion_Staking.update_distributed_reward(uint256)._value (Tyrion_Staking.sol#520) is not in mixedCase
Function Tyrion_Staking.update_Unstake_request_time(uint256) (Tyrion_Staking.sol#527-532) is not in mixedCase
Parameter Tyrion_Staking.update_Unstake_request_time(uint256)._value (Tyrion_Staking.sol#527) is not in mixedCase
Function Tyrion_Staking.update_APY(uint256) (Tyrion_Staking.sol#534-539) is not in mixedCase
Parameter Tyrion_Staking.update_APY(uint256)._value (Tyrion_Staking.sol#534) is not in mixedCase
Variable Tyrion_Staking.Staking_token (Tyrion_Staking.sol#20) is not in mixedCase
Variable Tyrion_Staking.Reward_Token (Tyrion_Staking.sol#21) is not in mixedCase
Variable Tyrion_Staking.Apy_Timeframe (Tyrion_Staking.sol#27) is not in mixedCase
Variable Tyrion_Staking.Lockup_period (Tyrion_Staking.sol#28) is not in mixedCase
Variable Tyrion_Staking.Apy (Tyrion_Staking.sol#29) is not in mixedCase
Variable Tyrion_Staking.minimum_Apy (Tyrion_Staking.sol#30) is not in mixedCase
Variable Tyrion_Staking.minimum_investment (Tyrion_Staking.sol#32) is not in mixedCase
Variable Tyrion_Staking.maximum_investment (Tyrion_Staking.sol#33) is not in mixedCase
Variable Tyrion_Staking.minimum_withdraw_reward_limit (Tyrion_Staking.sol#34) is not in mixedCase
Variable Tyrion_Staking.Unstake_request_time (Tyrion_Staking.sol#35) is not in mixedCase
Variable Tyrion_Staking.total_reward_to_distribute (Tyrion_Staking.sol#36) is not in mixedCase
Variable Tyrion_Staking.All_investors (Tyrion_Staking.sol#40) is not in mixedCase
Variable Tyrion_Staking.launch_time (Tyrion_Staking.sol#41) is not in mixedCase
Variable Tyrion_Staking.user_investments (Tyrion_Staking.sol#75) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Tyrion_Staking.slitherConstructorVariables() (Tyrion_Staking.sol#16-542) uses literals with too many digits:
        - maximum_investment = 100000 * 10 ** 9 (Tyrion_Staking.sol#33)
Tyrion_Staking.slitherConstructorVariables() (Tyrion_Staking.sol#16-542) uses literals with too many digits:
        - total_reward_to_distribute = 50000000 * 10 ** 9 (Tyrion_Staking.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Tyrion_Staking.Reward_Token (Tyrion_Staking.sol#21) should be constant
Tyrion_Staking.Staking_token (Tyrion_Staking.sol#20) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

Tyrion_Staking.launch_time (Tyrion_Staking.sol#41) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
Tyrion_Staking.sol analyzed (2 contracts with 84 detectors), 90 result(s) found
```

# Solidity Static Analysis

**Tyrion_Staking.sol**

### Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Tyrion_Staking.Stake(uint256,bool): Could potentially lead to re-entrancy vulnerability.

more

Pos: 89:8:

### Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Tyrion_Staking.withdrawReward(): Could potentially lead to re-entrancy vulnerability.

more

Pos: 316:8:

### Block timestamp:

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.

more

Pos: 225:33:

### Gas costs:

Gas requirement of function Tyrion_Staking.get_apy_temp is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)

Pos: 127:8:



**Email: audit@EtherAuthority.io**

### Gas costs:

Gas requirement of function Tyrion_Staking.getAll_investments is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 384:8:

### Gas costs:

Gas requirement of function Tyrion_Staking.getAll_investments_ForReward is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 414:8:

### Constant/View/Pure functions:

Token.transfer(address,uint256) : Potentially should be constant/view/pure but is not.
more
Pos: 10:4:

### Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 516:12:

# Solhint Linter

## Tyrion_Staking.sol

```
Compiler version ^0.8.0 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:6
Contract has 20 states declarations but allowed no more than 15
Pos: 1:15
Contract name must be in CamelCase
Pos: 1:15
Explicitly mark visibility of state
Pos: 9:19
Variable name must be in mixedCase
Pos: 9:19
Explicitly mark visibility of state
Pos: 9:20
Variable name must be in mixedCase
Pos: 9:20
Variable name must be in mixedCase
Pos: 9:26
Variable name must be in mixedCase
Pos: 9:27
Variable name must be in mixedCase
Pos: 9:28
Variable name must be in mixedCase
Pos: 9:29
Variable name must be in mixedCase
Pos: 9:31
Variable name must be in mixedCase
Pos: 9:32
Variable name must be in mixedCase
Pos: 9:33
Variable name must be in mixedCase
Pos: 9:34
Variable name must be in mixedCase
Pos: 9:35
Variable name must be in mixedCase
Pos: 9:39
Variable name must be in mixedCase
Pos: 9:40
Contract name must be in CamelCase
Pos: 9:41
Variable name must be in mixedCase
Pos: 13:45
Variable name must be in mixedCase
Pos: 13:50
Variable name must be in mixedCase
Pos: 13:51
Variable name must be in mixedCase
Pos: 13:52
Variable name must be in mixedCase
Pos: 13:64
```

```
Variable name must be in mixedCase
Pos: 13:74
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 9:76
Avoid making time-based decisions in your business logic
Pos: 25:79
Function name must be in mixedCase
Pos: 9:88
Avoid making time-based decisions in your business logic
Pos: 58:107
Avoid making time-based decisions in your business logic
Pos: 60:108
Function name must be in mixedCase
Pos: 9:126
Visibility modifier must be first in list of modifiers
Pos: 38:126
Variable name must be in mixedCase
Pos: 13:127
Function name must be in mixedCase
Pos: 9:165
Visibility modifier must be first in list of modifiers
Pos: 33:165
Variable name must be in mixedCase
Pos: 13:166
Function name must be in mixedCase
Pos: 9:207
Visibility modifier must be first in list of modifiers
Pos: 41:207
Variable name must be in mixedCase
Pos: 13:212
Avoid making time-based decisions in your business logic
Pos: 34:224
Function name must be in mixedCase
Pos: 9:262
Visibility modifier must be first in list of modifiers
Pos: 48:262
Variable name must be in mixedCase
Pos: 13:266
Avoid making time-based decisions in your business logic
Pos: 34:276
Variable name must be in mixedCase
Pos: 13:316
Error message for require is too long
Pos: 13:317
Possible reentrancy vulnerabilities. Avoid state changes after
transfer.
Pos: 13:320
Error message for require is too long
Pos: 13:331
Avoid making time-based decisions in your business logic
Pos: 119:335
Variable name must be in mixedCase
Pos: 17:337
Possible reentrancy vulnerabilities. Avoid state changes after
transfer.
Pos: 13:343
Possible reentrancy vulnerabilities. Avoid state changes after
transfer.
```

```
Pos: 13:344
Avoid making time-based decisions in your business logic
Pos: 59:344
Possible reentrancy vulnerabilities. Avoid state changes after
transfer.
Pos: 13:346
Possible reentrancy vulnerabilities. Avoid state changes after
transfer.
Pos: 13:347
Function name must be in mixedCase
Pos: 9:356
Error message for require is too long
Pos: 13:360
Avoid making time-based decisions in your business logic
Pos: 64:363
Avoid making time-based decisions in your business logic
Pos: 67:364
Function name must be in mixedCase
Pos: 9:372
Function name must be in mixedCase
Pos: 9:383
Variable name must be in mixedCase
Pos: 60:383
Function name must be in mixedCase
Pos: 9:413
Variable name must be in mixedCase
Pos: 70:413
Error message for require is too long
Pos: 13:433
Function name must be in mixedCase
Pos: 9:437
Visibility modifier must be first in list of modifiers
Pos: 48:437
Variable name must be in mixedCase
Pos: 13:440
Function name must be in mixedCase
Pos: 9:446
Avoid making time-based decisions in your business logic
Pos: 20:448
Function name must be in mixedCase
Pos: 9:451
Provide an error message for require
Pos: 13:459
Provide an error message for require
Pos: 13:462
Function name must be in mixedCase
Pos: 9:471
Provide an error message for require
Pos: 13:473
Function name must be in mixedCase
Pos: 9:478
Provide an error message for require
Pos: 13:480
Function name must be in mixedCase
Pos: 9:485
Provide an error message for require
Pos: 13:487
Function name must be in mixedCase
Pos: 9:492
```

```
Provide an error message for require
Pos: 13:494
Function name must be in mixedCase
Pos: 9:499
Provide an error message for require
Pos: 13:501
Function name must be in mixedCase
Pos: 9:506
Provide an error message for require
Pos: 13:508
Function name must be in mixedCase
Pos: 9:513
Provide an error message for require
Pos: 13:515
Function name must be in mixedCase
Pos: 9:519
Provide an error message for require
Pos: 13:521
Function name must be in mixedCase
Pos: 9:526
Provide an error message for require
Pos: 13:528
Function name must be in mixedCase
Pos: 9:533
Provide an error message for require
Pos: 13:535
```

**Software analysis result:**

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.




**Email: audit@EtherAuthority.io**