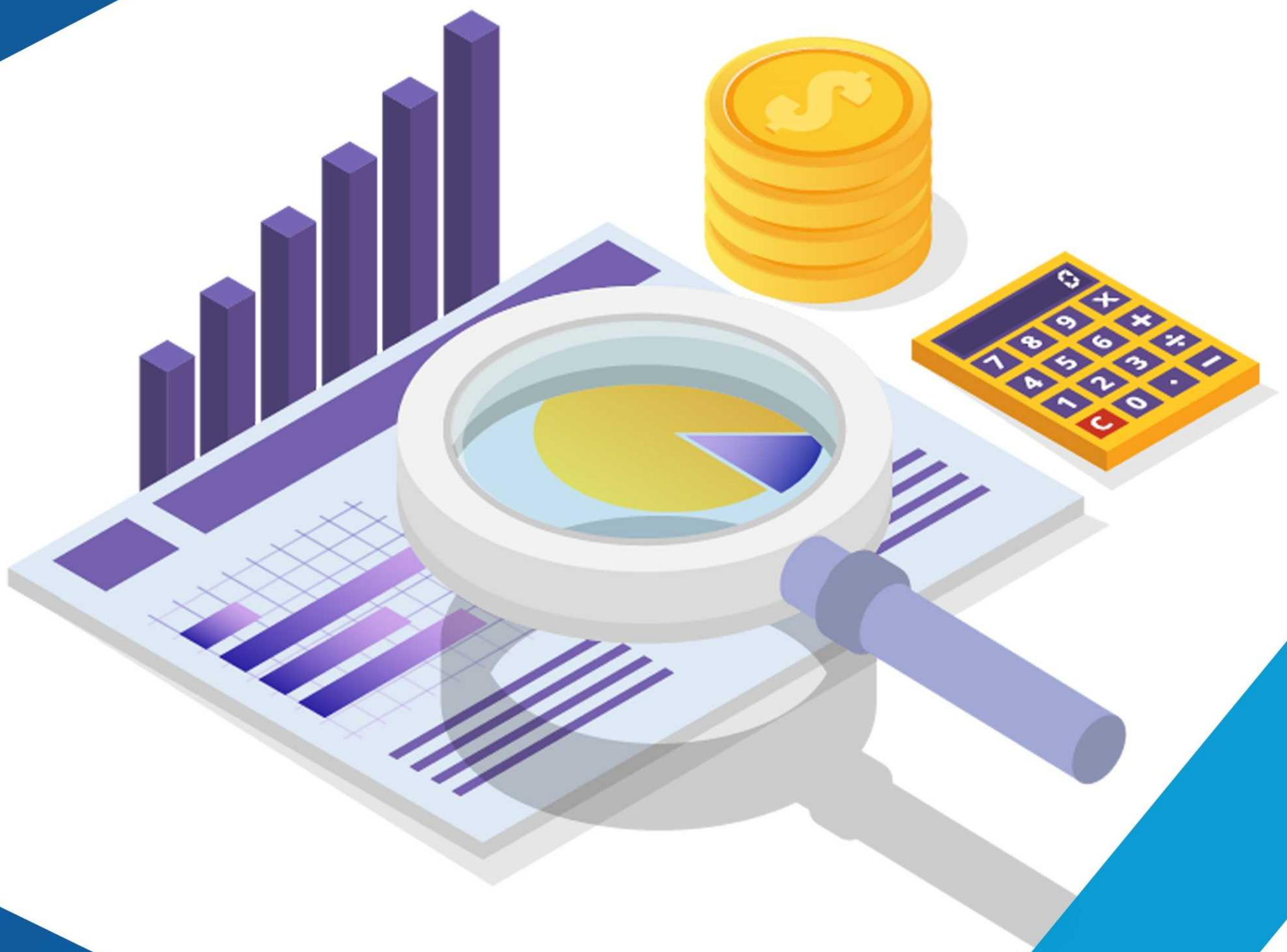
QuillAudits



Audit Report
April, 2021

Contents

Overview

TrustPad

TrustPad is a decentralized multi-chain fundraising platform enabling projects to raise capital and promise safety to early-stage investors.

Name: TrustPad

Symbol: TPAD

Decimals: 9

2% re-distribution fee on each transaction

Contract: TrustPad.sol

Description Report: TrustPad.md

Scope of Audit

The scope of this audit was to analyse **TrustPad.sol** smart contract's codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Malicious libraries
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- ERC20 transfer() does not return boolean
- ERC20 approve() race
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low level calls
- Missing Zero Address Validation

- Private modifier
- Revert/require functions
- Using block.timestamp
- Multiple Sends
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.
SmartCheck.

Static Analysis
Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to test security of smart contracts.

Code Review / Manual Analysis
Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of automated analysis were manually verified.

Gas Consumption
In this step we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Ganache, Solhint, Mythril, Slither, SmartCheck.

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity and each of them has been explained below.

High severity issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

Medium level severity issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems and they should still be fixed.

Low level severity issues

Low level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity four issues which indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	1	0	0
Closed	0	0	1	5

Issues Found – Code Review / Manual Testing

High severity issues

None.

Medium severity issues

1. **ERC20 approve() race**

   The standard ERC20 implementation contains a widely-known racing condition in its approve function, wherein a spender is able to witness the token owner broadcast a transaction altering their approval, and quickly sign and broadcast a transaction using transferFrom to move the current approved amount from the owner's balance to the spender. If the spender's transaction is validated before the owner's, the spender is able to spend their entire approval amount twice.

   Possible Solution: make sure to create user interfaces in such a way that they set the allowance first to 0 before setting it to another value for the same spender

   Reference:

   - https://docs.google.com/document/d/1YLPtQxZu1UAvO9cZ1O2RPXBbT0mooh4DYKjA_jp-RLM/edit
   - https://medium.com/mycrypto/bad-actors-abusing-erc20-approval-to-steal-your-tokens-c0407b7f7c7c
   - https://eips.ethereum.org/EIPS/eip-20

Low level severity issues

[FIXED]SafeMath library not imported.
Note: If the contract is going to be compiled with solc >=0.8.0. There is no need to import SafeMath library, SafeMath is already integrated with solc>=0.8.0. Operators automatically revert on overflows

Informational

1. **[FIXED] [#L3]** Incorrect Versions of Solidity: **Pragma version>=0.7.0** allows old versions, Using an old version prevents access to new Solidity security checks. We recommend using the latest version of solidity.

2. **[FIXED] [#L140]** A check can be added in the function **excludeAccount**, to avoid excluding Contract Address itself.
   Recommendation: **require(account != address(this), “Cannot exclude self contract”)**

3. **[FIXED] [#L149-160]** function **includeAccount** and function **_getCurrentSupply [#L270-280]** consuming Extra Gas: **.length** of non-memory array is used in the condition for loop. In this case, every iteration of the loop consumes extra gas. Holding its value in a local variable is more gas efficient. Also if array.length is large enough, the function can exceed the block gas limit. So, it is recommended to avoid loops with an unknown number of steps

4. **[FIXED] [#L150]** It should be **included** instead of **excluded** in the require function

5. **[FIXED]**
   [#L163]function setFeeless(address account, bool **isFeeless**),
   [#L241]function _getValues(uint256 tAmount, bool **isFeeless**),
   [#L248]function _getTValues(uint256 tAmount, bool **isFeeless**),
   Shadows [#L106] function **isFeeless**(address account)

Gas Optimization

Public functions that are never called by the contract should be declared external to save gas.

```
name() should be declared external:
        - ERC20.name() (ERC20.sol#60-62)
symbol() should be declared external:
        - ERC20.symbol() (ERC20.sol#68-70)
decimals() should be declared external:
        - ERC20.decimals() (ERC20.sol#85-87)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (ERC20.sol#99-101)
        - TrustPad.balanceOf(address) (trustpad.sol#68-71)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (ERC20.sol#111-114)
        - TrustPad.transfer(address,uint256) (trustpad.sol#73-76)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (ERC20.sol#119-121)
        - TrustPad.allowance(address,address) (trustpad.sol#78-80)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (ERC20.sol#130-133)
        - TrustPad.approve(address,uint256) (trustpad.sol#82-85)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (ERC20.sol#148-156)
        - TrustPad.transferFrom(address,address,uint256) (trustpad.sol#87-91)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (ERC20.sol#170-173)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (ERC20.sol#189-195)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Ownable.sol#54-57)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Ownable.sol#63-67)
name() should be declared external:
        - TrustPad.name() (trustpad.sol#52-54)
symbol() should be declared external:
        - TrustPad.symbol() (trustpad.sol#56-58)
decimals() should be declared external:
        - TrustPad.decimals() (trustpad.sol#60-62)
increaseAllowance(address,uint256) should be declared external:
        - TrustPad.increaseAllowance(address,uint256) (trustpad.sol#93-96)
decreaseAllowance(address,uint256) should be declared external:
        - TrustPad.decreaseAllowance(address,uint256) (trustpad.sol#98-101)
isExcluded(address) should be declared external:
        - TrustPad.isExcluded(address) (trustpad.sol#103-105)
isFeeless(address) should be declared external:
        - TrustPad.isFeeless(address) (trustpad.sol#107-109)
totalFees() should be declared external:
        - TrustPad.totalFees() (trustpad.sol#111-113)
reflect(uint256) should be declared external:
        - TrustPad.reflect(uint256) (trustpad.sol#115-122)
reflectionFromToken(uint256,bool) should be declared external:
        - TrustPad.reflectionFromToken(uint256,bool) (trustpad.sol#124-133)
```

Automated Testing

Slither

Slither didn’t detect any high severity issues

Mythril

Mythril didn’t detect any high severity issues

Smartcheck

Smartcheck didn’t detect any high severity issues

Solhint

```
trustpad.sol
   88:2  error  Line length must be no more than 120 but current length is 130  max-line-length
   98:2  error  Line length must be no more than 120 but current length is 138  max-line-length
  194:2  error  Line length must be no more than 120 but current length is 139  max-line-length
  203:2  error  Line length must be no more than 120 but current length is 139  max-line-length
  213:2  error  Line length must be no more than 120 but current length is 139  max-line-length
  223:2  error  Line length must be no more than 120 but current length is 139  max-line-length
  241:2  error  Line length must be no more than 120 but current length is 125  max-line-length
  258:2  error  Line length must be no more than 120 but current length is 127  max-line-length

✖ 8 problems (8 errors, 0 warnings)
```

Disclaimer

The audit does not give any warranties on the security of the code. One audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of the code. Besides, a security audit, please don’t consider this report as investment advice.

Closing Summary

Several issues of medium and low severity have been reported during the audit, out of which, most of them have been fixed. Some suggestions have also been made to improve the code quality and gas optimization. There were NO critical or major issues found that can break the intended behavior.

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

hello@quillhash.com