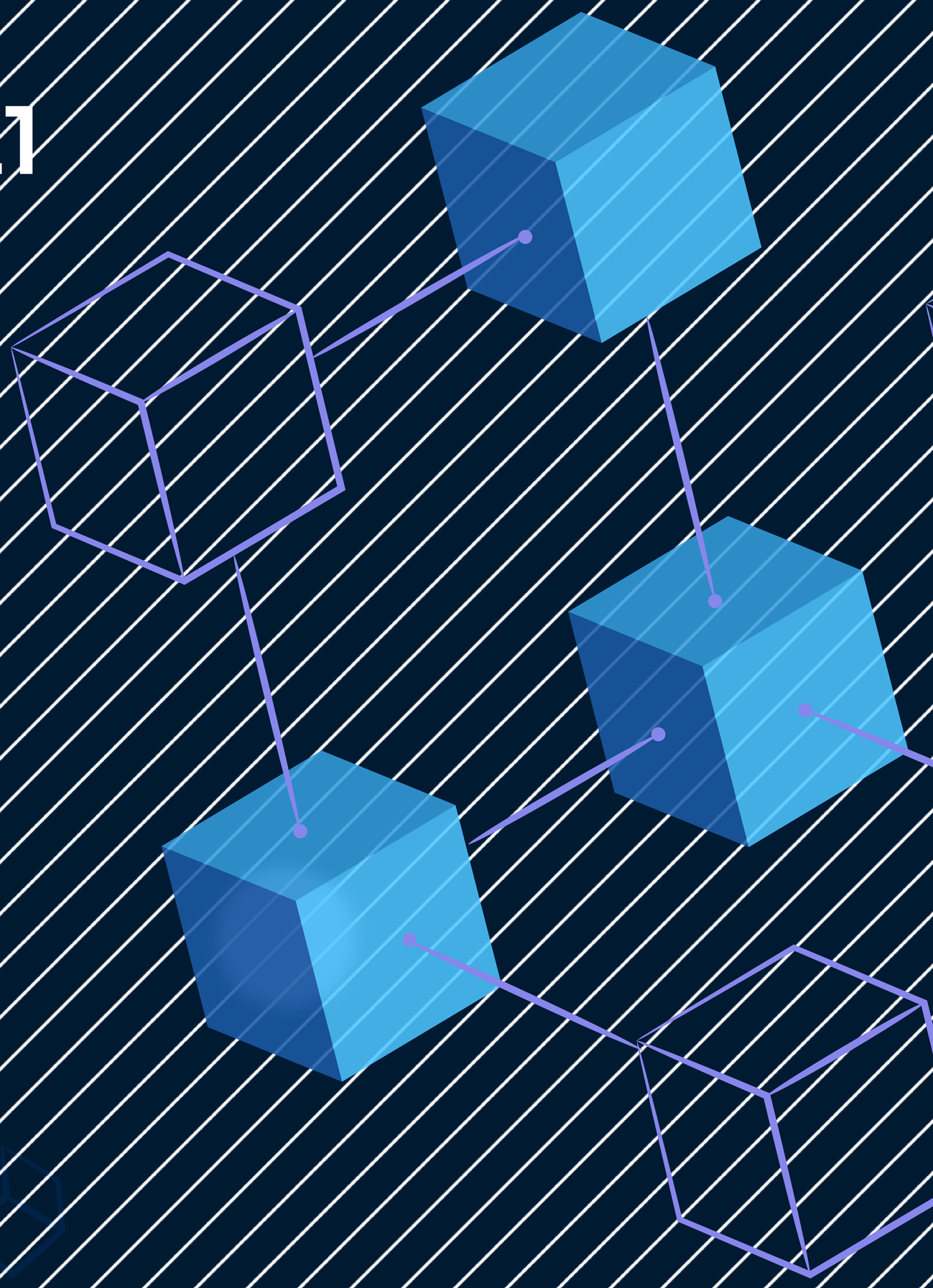
QuillAudits

# Audit Report
# November, 2021



For

# Contents

## Scope of the Audit

The scope of this audit was to analyze and document the T99(TNN) Token smart contract 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
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- BEP20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

# 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 BEP-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 analysed 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.

**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 the security of smart contracts.

**Code Review / Manual Analysis**
Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the 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 the possibilities of optimization of code to reduce gas consumption.

**Tools and Platforms used for Audit**
Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis, Theo.

# Issue Categories

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

| Risk-level | Description |
|---|---|
| High | 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 be fixed before moving to a live environment. |
| Medium | 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 | 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 issues that 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 | 0 | 0 | 1 |
| Acknowledged | 0 | 0 | 0 | 0 |
| Closed | 3 | 2 | 1 | 1 |

# Introduction

**Nov 10, 2021 - Nov 21,2021** QuillAudits Team performed a security audit for an BEP20 token smart contract - **T99(TNN)** Token.

The code for the audit was taken from the following official link: https://bscscan.com/address/0x82a3cd29baab1523a011b442ed37e9a86f824b6e#code

| V | Date | Commit hash | Note |
|---|---|---|---|
| 1 | November | https://bscscan.com/address/0x82a3cd29baab1523a011b442ed37e9a86f824b6e#code | |
| 2 | November | https://docs.google.com/document/d/12qaWXg7qHySOObTacHKqD8VxTJdCxxmWgyEd-0DbsaQ/edit?usp=sharing | T99_Fixed.sol |
| 3 | November | https://docs.google.com/document/d/1tw7EVihpIuvf-Y5y5fTiisIP3wm34VOkpI_8t9qFDxA/edit?usp=sharing | T99_Fixed2.sol |
| 4 | November | https://bscscan.com/address/0xB4a8Aed9b705D780B29D48cCa2A6Ca9d63feaf14#code | |

# Issues Found

## A. Contract – T99(TNN)Token

### High severity issues

1. Blacklist can be bypassed with the help of approve and transferFrom functionalities

| Line | Code |
| --- | --- |
| 146-164 | function transferFrom(address _from, address _to, uint256 _value) public returns (bool) {<br>require(tokenBlacklist[msg.sender] == false);<br>require(_to != address(0));<br>require(_value <= balances[_from]);<br>require(_value <= allowed[_from][msg.sender]);<br><br>balances[_from] = balances[_from].sub(_value);<br>balances[_to] = balances[_to].add(_value);<br>allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);<br>emit Transfer(_from, _to, _value);<br>return true;<br>}<br><br>function approve(address _spender, uint256 _value) public returns (bool) {<br>allowed[msg.sender][_spender] = _value;<br>emit Approval(msg.sender, _spender, _value);<br>return true;<br>} |

**Description**
Any blacklisted user can pass the check by using approve and transferFrom functions.

**Remediation**
Fix require statement **require(tokenBlacklist[from] == false);** in **transferFrom** function. Document the extent and requirements of blacklist functionality.

**Status:** Closed. Issues has been fixed

2. **Token supply is not fixed**

**Description**
Project's specifications mention a fixed total supply, however it can be increased at the behest of the contract's owner and decreased on the behest of the token's owner.

**Remediation**
Ensure either specification is in synchronicity with contract or vice-versa.

**Status:** Closed. Issues has been fixed

3. **The blacklistAddress function has been omitted**

**Description**
After the changes made to the contract after the initial audit suggestions, the PausableToken contract was removed altogether so the blacklistAddress function was omitted. The internal _blacklist function is still present but not used anywhere.

**Remediation**
Make an external or public blacklistAddress function in which you can call the internal _blacklist function.

**Status:** Closed. Issues has been fixed

## Medium severity issues

### 4. Improper error handling

**Description**

Across the codebase, appropriate arithmetic calculations do not lead to vulnerability but they revert to unexpected values, so if a transaction is reverted due to invalid, the user will not know the reason for the failure. E.g. Using **transferFrom** with more than allowance.

**Remediation**

Properly analyze all possible conditions in which a transaction could fail and check them explicitly with appropriate error messages.

**Status:** Closed. Issues has been fixed

### 5. Tokens can be minted or burned even if contract is paused

| Line | Code |
|---|---|
| 248-266 | function burn(uint256 _value) public {<br>_burn(msg.sender, _value);<br>}<br><br>function _burn(address _who, uint256 _value) internal {<br>require(_value <= balances[_who]);<br>balances[_who] = balances[_who].sub(_value);<br>totalSupply = totalSupply.sub(_value);<br>emit Burn(_who, _value);<br>emit Transfer(_who, address(0), _value);<br>}<br><br>function mint(address account, uint256 amount) onlyOwner public {<br><br>totalSupply = totalSupply.add(amount);<br>balances[account] = balances[account].add(amount);<br>emit Mint(address(0), account, amount);<br>emit Transfer(address(0), account, amount);<br>} |

**Description**
Most token functionality is restricted when the contract is paused, however there is no restriction on **mint** and **burn** functions.

**Remediation**
If this is intended, then specifications should mention this otherwise restrict **mint** and **burn** functions.

**Status:** Closed. Issues has been fixed

## Low severity issues

### 6. Pragma version not fixed

**Description**
It is a good practice to lock the solidity version for a live deployment (use **0.8.0** instead of **^0.8.0**). Contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs. Contracts may also be deployed by others and the pragma indicates the compiler version intended by the original authors.

**Remediation**
Remove the ^ sign to lock the pragma version.

**Status:** Closed. Issues has been fixed

### 7. Compiler version is too old

**Description**
The compiler being used was released 3-4 years ago. It’s recommended to use more recent compiler version, there can be benefits like reduction in bytecode size etc.

**Status:** Closed. Issues has been fixed

## Informational issues

### 8. Lack of unit tests

**Description**
The code is taken from **BscScan**, so we are under the assumption that there are no unit tests for the codebase. Contracts meant to handle thousands of user funds should have appropriate testing and coverage.

**Remediation**
Implement unit tests with at least 98% coverage.

**Status:** Open

# Automated Tests

## Slither

No major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorized above according to their level of severity.

# Closing Summary

Overall, in this audit, there are few critical severity issues associated with token transfers, which has been fixed by T99 Team.
No instances of Integer Overflow and Underflow vulnerabilities are found in the contract.

# Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the **T99(TNN) Token**. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the **T99(TNN) Token** team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

# Audit Report
# November, 2021

For

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com