VIDMA

POPNETWORK

SMART CONTRACT AUDIT

Project: POP Network
Date: July 15th, 2021

TABLE OF CONTENTS

SUMMARY

Vidma team has conducted a smart contract audit for the given codebase.

Within the manual part of the audit one critical issue was found. The issue is described in this final report and was successfully fixed by POP Network team. The total amount of issues found and their severity level is shown on the table below.

Despite the findings, we can state that the contract is in good condition and no security objections left after re-audit and review of the fixes. Based on the aforementioned information, we can conclude that the contracts are fully operational and secure for an end-user.

Severity of the issue	Total found	Resolved	Unresolved
Critical	1 issue	1 issue	0 issues
High	0 issues	0 issues	0 issues
Medium	1 issue	1 issue	0 issues
Low	1 issue	1 issue	0 issues
Informational	1 issue	1 issue	0 issues
Total	**4 issues**	**4 issues**	**0 issues**

Evaluating the findings, we can assure that the contract is safe to use and all the issues found are performed only by certain conditions and cases. Under the given circumstances we can set the following risk level:

High Confidence

Vidma auditors are evaluating the initial commit given for the scope of the audit and the last commit with the fixes. Hence, it helps to adequately evaluate the development quality.

Based on the given findings, risk level, performance, and code style, Vidma team can grant the following overall score:

Vidma auditing team has conducted a bunch of integrated autotests to ensure that the given codebase has decent performance and security levels. The test results and the coverage can be found in the accompanying section of this audit report.

Please mind that this audit does not certify the definite reliability and security level of the contract. This document describes all vulnerabilities, typos, performance issues, and security issues found by Vidma auditing team. If the code is under development, we recommend run one more audit once the code is finalized.

SCOPE OF WORK

POPNETWORK

POP Network is a user-friendly ecosystem of blockchain and AI applications built to power the streaming economy. Their vision is a universal system for turning attention into real-world goods and services.

Within the scope of this audit, two independent auditors deeply investigated the given codebase and analyzed the overall security and performance of smart contracts.

The debrief took place on July 15th, 2021 and the final results are present in this document.

Vidma auditing team has made a review of the following contracts:

- TokenRecover;
- POP;
- BEP20.

The source code was taken from the following **sources**:

- https://github.com/popnetwork/pop-network-token-bep20
- https://github.com/popnetwork/pop-network-token-erc20

Initial commit submitted for the audit:

- 404ec3ae6cea10fe777958eb38e1113a3ada5243
- b81f49ad6b358f387f3b59618e978b989dfc999f

Last commit:
404ec3ae6cea10fe777958eb38e1113a3ada5243

WORKFLOW OF THE AUDITING PROCESS

During the manual phase of the audit, Vidma team manually looks through the code in order to find any security issues, typos, or discrepancies with the logic of the contract.

Within the testing part, Vidma auditors run integration tests using the Truffle testing framework. The test coverage and the tests themselves are inserted into this audit report.

Vidma team uses the most sophisticated and contemporary methods and techniques to ensure the contract does not have any vulnerabilities or security risks:

- Re-entrancy;
- Access Management Hierarchy;
- Arithmetic Over/Under Flows;
- Unexpected Ether;
- Delegatecall;
- Default Public Visibility;
- Hidden Malicious Code;
- Entropy Illusion (Lack of Randomness);
- External Contract Referencing;
- Short Address/Parameter Attack;
- Unchecked CALL Return Values;
- Race Conditions / Front Running;
- General Denial Of Service (DOS);
- Uninitialized Storage Pointers;
- Floating Points and Precision;
- Tx.Origin Authentication;
- Signatures Replay;
- Pool Asset Security (backdoors in the underlying ERC-20).

STRUCTURE AND ORGANIZATION OF THE FINDINGS

For the convenience of reviewing the findings in this report, Vidma auditors classified them in accordance with the severity of the issues. (from most critical to least critical). The acceptance criteria are described below.

All issues are marked as "Resolved" or "Unresolved", depending on whether they have been fixed by POP Network or not. The latest commit, indicated in this audit report should include all the fixes made.

To ease the explanation, the Vidma team has provided a detailed description of the issues and recommendations on how to fix them.

Hence, according to the statements above, we classified all the findings in the following way:

Finding	Description
Critical	The issue bear a definite risk to the contract, so it may affect the ability to compile or operate.
High	Major security or operational risk found, that may harm the end-user or the overall performance of the contract.
Medium	The issue affects the contract to operate in a way that doesn't significantly hinder its performance.
Low	The found issue has a slight impact on the performance of the contract or its security.
Informational	The issue does not affect the performance or security of the contract/recommendations on the improvements.

MANUAL REPORT

The owner can burn user tokens to any address

Critical | Resolved

Function *burn* allows the owner of the contract to burn tokens from any user without user allowance.

```
function burn(address _to, uint256 _amount) public onlyOwner {
    _burn(_to, _amount);
}

function _burn(address account, uint256 amount) internal {
    require(account != address(0), 'BEP20: burn from the zero
address');
    _balances[account] = _balances[account].sub(amount, 'BEP20: burn
amount exceeds balance');
    _totalSupply = _totalSupply.sub(amount);
        emit Transfer(account, address(0), amount);
}
```

Recommendation:

Allow the token owner to burn tokens he own:

```
function burn(uint256 _amount) public {
        _burn(_msgSender(), _amount);
}
```

Only owner can be minter

Medium | Resolved

In contract POP.sol function *mint()* can be called only by the owner, but the owner can set the minter role for another wallet address by calling function *setMinter()*.

```
function mint(address _to, uint256 _amount) public onlyOwner {
    require(minter == msg.sender, "POP: caller is not a minter");
    require(_amount.add(totalSupply()) <= MAX_SUPPLY, "POP: maxcap
reached");
    _mint(_to, _amount);
}
```

Recommendation:

Remove OnlyOwner from the function *mint()*.

Unused function

Low | Resolved

The contract BEP20.sol has an internal function *_burnFrom()* that is not used/ called in the contract POP.sol

Recommendation:

You can remove function *_burnFrom()* from BEP20.sol if it is not needed or create a new function in contract POP.sol which will call it.

Mismatch between function name and expected function behavior

Informational | Resolved

In contract TokenRecover.sol function for recovering BEP20 tokens named as *recoverERC20*.

Recommendation:

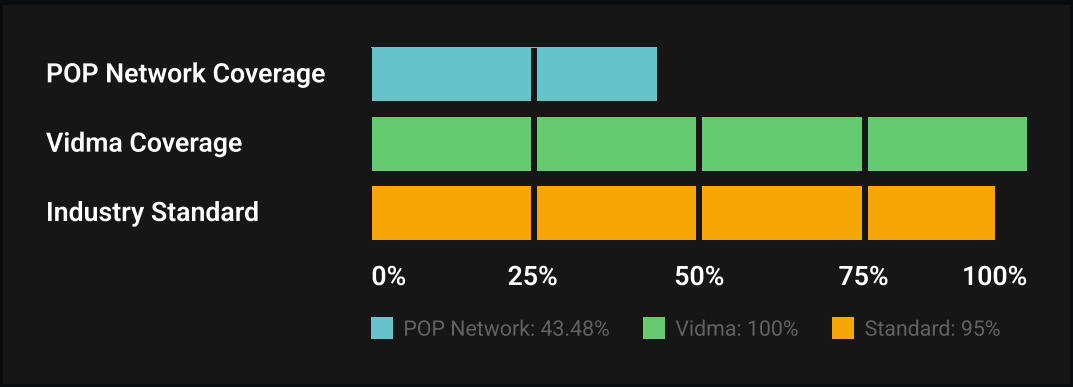
You can change the name of this function to recoverBEP20 or simply *tokenRecover*.

TEST RESULTS

To verify the contract security and performance a bunch of integration tests were made using the Truffle testing framework.

Tests were based on the functionality of the code, business logic, and requirements and for the purpose of finding the vulnerabilities in the contacts.

In this section, we provide both tests written by POP Network and Vidma auditors.



It's important to note that Vidma auditors do not modify, edit or add tests to the existing tests provided in the POP Network repo. We write totally separate tests with code coverage of a minimum of 95%, to meet the industry standards.

Tests are written by POP Network

Test Coverage

File	% Stmts	% Branch	% Funcs	% Lines
contracts\	43.48	31.25	36.36	43.48
BEP20.sol	42.11	25.00	35.29	42.11
POP.sol	57.14	50.00	50.00	57.14
TokenRecover.sol	0	100.00	0	0
All files	**43.48**	**31.25**	**36.36**	**43.48**

Test Results

```
Contract: POP contract
  Deployment
    ✓ Should set the right owner (185ms)
    ✓ Should assign the total supply of tokens to be
      the owner (498ms)
  Transactions
    ✓ Should transfer tokens between accounts (2100ms)

3 passing (6s)
```

Tests are written by Vidma

Test Coverage

File	% Stmts	% Branch	% Funcs	% Lines
contracts\	100.00	94.44	100.00	100.00
contract.sol	100.00	95.95	100.00	100.00
BEP20.sol	100.00	83.33	100.00	100.00
POP.sol	100.00	100.00	100.00	100.00
TokenRecover.sol	100.00	100.00	100.00	100.00
All files	**100.00**	**94.44**	**100.00**	**100.00**

Test Results

```
Contract: POP ERC-20
  POP Test Cases
    ✓ should deploy with correct owner (98ms)
    ✓ should deploy with correct name (135ms)
    ✓ should deploy with correct symbol (111ms)
    ✓ should deploy with correct decimals (159ms)
    ✓ should deploy with correct max supply (131ms)
    ✓ should deploy with correct total supply (210ms)
    ✓ should deploy with correct owner balance (805ms)
    ✓ shouldn't deploy with cap setted to zero (960ms)
    ✓ should transfer tokens correctly (806ms)
    ✓ shouldn't transfer tokens to the zero address (221ms)
    ✓ shouldn't transfer tokens if transfer amount exceed
      balance (461ms)
```

```
    ✓ should approve correctly (455ms)
    ✓ shouldn't approve to the zero address (239ms)
    ✓ should increase allowance correctly (861ms)
    ✓ shouldn't increase allowance for zero address (317ms)
    ✓ should decrease allowance correctly (714ms)
    ✓ shouldn't decrease allowance for zero address (289ms)
    ✓ shouldn't decrease allowance below zero (251ms)
    ✓ should transfer tokens from address correctly (1168ms)
    ✓ shouldn't transfer tokens from address if amount exceed
      allowance (267ms)
    ✓ should burn tokens correctly (619ms)
    ✓ shouldn't burn tokens if burn amount exceed balance (250ms)
    ✓ should burn from address correctly (1170ms)
    ✓ should burn from address correctly (1162ms)
    ✓ shouldn't burn from address if burn amount exceed
      allowance (868ms)
    ✓ should mint tokens correctly (776ms)
    ✓ shouldn't mint tokens by not the owner (606ms)
    ✓ shouldn't mint tokens if max cap reached (279ms)
    ✓ shouldn't mint tokens to the zero address (653ms)
    ✓ should add new minter role correctly (1626ms)
    ✓ should renounce minter correctly (1250ms)
    ✓ should renounce ownership correctly (178ms)
    ✓ should transfer ownership correctly (308ms)
    ✓ shouldn't transfer ownership to the zero address (278ms)
    ✓ should recover ERC20 tokens correctly (2033ms)
    ✓ shouldn't recover ERC20 tokens by not the owner (1647ms)
    ✓ should query if a contract implements an interface
      correctly (1535ms)
    ✓ should transfer and call correctly (2613ms)
    ✓ should approve and call correctly (719ms)
    ✓ should safe math work correctly (1417ms)

Contract: POP BER-20
  POP Test Cases
    ✓ should deploy with correct owner (439ms)
    ✓ should deploy with correct name (402ms)
    ✓ should deploy with correct symbol (183ms)
    ✓ should deploy with correct decimals (258ms)
    ✓ should deploy with correct max supply (122ms)
    ✓ should deploy with correct minter (121ms)
    ✓ should set new minter correctly (419ms)
```

```
    ✓ shouldn't set minter by not the owner (663ms)
    ✓ should transfer tokens correctly (660ms)
    ✓ shouldn't transfer tokens to the zero address (279s)
    ✓ shouldn't transfer tokens if transfer amount exceed
      balance (237ms)
    ✓ shouldn't transfer from zero address (226ms)
    ✓ should approve correctly (368ms)
    ✓ shouldn't approve to the zero address (214ms)
    ✓ should increase allowance correctly (645ms)
    ✓ should decrease allowance correctly (687ms)
    ✓ shouldn't decrease allowance below zero (276ms)
    ✓ should transfer tokens from address correctly (714ms)
    ✓ should burn tokens correctly (525ms)
    ✓ shouldn't burn tokens if burn amount exceed balance (330ms)
    ✓ should mint tokens correctly (793ms)
    ✓ shouldn't mint tokens by not the minter (570ms)
    ✓ shouldn't mint tokens if max cap reached (296ms)
    ✓ shouldn't mint tokens to the zero address (493ms)
    ✓ should recover tokens correctly (1412ms)

  65 passing (3m)
```

We are delighted to have a chance to work together with POP Network team and contribute to their success by reviewing and certifying the security of the 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.

Website: **vidma.io**
Email: **security@vidma.io**

VIDMA