VIDMA

POPNETWORK

SMART CONTRACT AUDIT

Project: POP Network
Date: June 23, 2021

TABLE OF CONTENTS

SUMMARY

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

The contract is well written and structured. Based on the findings we can conclude that the contract is fully production-ready.

During the auditing process, Vidma team has found several issues, including issues with critical severity. Mostly every issue was resolved by the POP Network team. The only issue that is still unresolved has an informational level of severity and does not affect the security or performance side of the contract.

Severity of the issue	Total found	Resolved	Unresolved
Critical	2 issues	2 issues	0 issues
High	1 issue	1 issue	0 issues
Medium	2 issues	2 issues	0 issues
Low	1 issue	1 issue	0 issues
Informational	2 issues	1 issue	1 issue
Total	**8 issues**	**7 issues**	**1 issue**

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:

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 Jun 23rd, 2021 and the final results are present in this document

Vidma auditing team has made a review of the following contract: PopStaking.

The source code was taken from the following **source**: https://github.com/popnetwork/pop-staking-smart-contract/

Last commit: 94ff45f72cf4bf95d04e3ef4a41c0eacd4a0fbed

To conduct a more detailed audit, POP Network has provided the following **documentation**: https://docs.google.com/document/d/1tgg_rYItXD3_lBKJYe1B6rKS1vbQRy610eRn2QAhYu8/edit

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

Unknown behavior of *updatePendingInfo* function

Critical | Resolved

Entire contract depends on this function, but little or no details were provided on how often this func will be called, what conditions, what limits of each user, how many users, etc.

Additionally, this function actually contains two actions: users pending info update and pop per block calculation.

Worth to mention, that right now *devaddr* has too much power.

Recommendation:

if this is expected behavior (for example *davaddr* is a contract), please provide as many details as possible how users' funds are protected and stake pool is fair. And it would be better to do pop per block calculation "on the fly", more details on this in next issue.

Re-Audit:

Pop masternode running time is logged in backend and backend cron job will call *updatePendingInfo* function everyday with the updated Info with dev address permission.

The owner can transfer user tokens to any address

Critical | Resolved

Function *tokenTransfer* allows the owner of the contract to transfer all pop tokens from the balance of the staking contract to any address.

```
function tokenTransfer(address _to, IERC20 _token, uint256 _amount)
public onlyOwner {
    uint256 tokenBal = _token.balanceOf(address(this));
    if (_amount > tokenBal) {
        _token.transfer(_to, tokenBal);
    } else {
        _token.transfer(_to, _amount);
    }
}
```

Recommendation:

Set these tokens to be not salvageable.

Re-Audit:

Fixed.

popPerBlock may not be updated in time

High | Resolved

It is updated only on *updatePendingInfo* call. If, for any reason, the call does not happen in time – *popPerBlock* becomes outdated.

Additionally:

- Before cycle 1 *popPerBlock* value is 0, which will reset claimable value to 0 due to this multiplication: *user.amount.mul(popPerBlock)*;
- After cycle four there is no reset of the *popPerBlock* variable to a default value. So it will get stuck on *popPerBlockCycleFour* value (unless this is requirement).

Recommendation:

Do calculation of *popPerBlock* automatically, example:

```
function getPopPerBlock() public view returns (uint256) {
    //pass & save uint256[] _popPerBlockAllCycles; in constructor
    if(block.timestamp < startTime)return 1;
    uint256 cycle = (block.timestamp - startTime) / 90 days;
    //we can hardcode 3 to save gas
    (_popPerBlockAllCycles.length - 1)
    if(cycle > 3)cycle = 3;
    return _popPerBlockAllCycles[cycle];
}
```

Re-Audit:

Fixed.

Constructor could be improved

Medium | Resolved

There are several options of improvements, based on *popPerBlock* calculation improve.

Recommendation:

Depending on deployments count:

- If only one pool: you can simply set *popPerBlockCycleOne...Four* as constants, and set *popPerBlock* on initialization;
- If several pools, then you need to have that calculation in one place. Please create *getPopPerBlock()* (example of it shown above) and remove what is not needed from the constructor.

Re-Audit:

Fixed.

No check for 0 value

Medium | Resolved

Functions *deposit* / *withdraw* have no check for an amount equal to 0.

Recommendation:

Functions *deposit* / *withdraw* have no check for an amount equal to 0.

```
require(amount > 0, " deposit: can't deposit 0");
require(amount > 0, "withdraw: can't withdraw 0");
```

Re-Audit:

Fixed.

Unnecessary variable initialization

Low | Resolved

In constructor at line 61 no need to initialize *popPerBlock* variables, since it's already been initialized by default.

Recommendation:

Remove line #61:

```
popPerBlock = 0;
```

Re-Audit:

Fixed.

No practical use in function *getMultiplier*

Informational | Unresolved

Function *getMultiplier* simply subtracts one number from another. It has no practical use for the end-user.

Recommendation:

It should return user reward multiplier instead

```
function getMultiplier() public view returns (uint256) {
    UserInfo storage user = userInfo[msg.sender];
    return user.rewardMultiplier
}
```

No SPDX-License-Identifier found

Informational | Resolved

Trust in smart contracts can be better established if their source code is available. Since making source code available always touches on legal problems with regards to copyright, the Solidity compiler encourages the use of machine-readable SPDX license identifiers.

Recommendation:

Every source file should start with a comment indicating its license.

Example:

```
// SPDX-License-Identifier: UNLICENSED
```

Re-Audit:

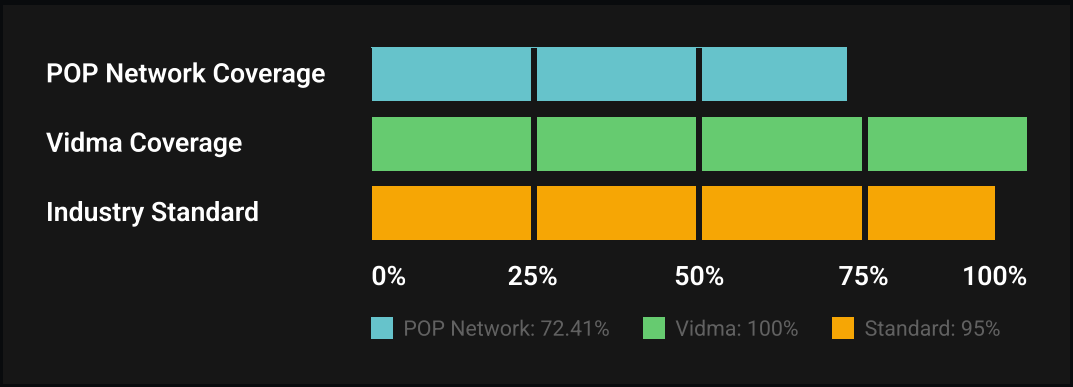
Fixed.

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\	72.41	45.45	70.00	75.44
PopStaking.sol	72.41	45.45	70.00	75.44
All files	**72.41**	**45.45**	**70.00**	**75.44**

Test Results

```
PopStaking contract
  Deployment
    ✓ Should set the right owner
    ✓ Should set correct state variables
  Staking
    ✓ should give out POPs only after staking time - 1 (301ms)
    ✓ should give out POPs only after staking time - 2 (253ms)

4 passing (3s)
```

Tests are written by Vidma

Test Coverage

File	% Stmts	% Branch	% Funcs	% Lines
contracts\	100.00	100.00	100.00	100.00
PopStaking.sol	100.00	100.00	100.00	100.00
All files	**100.00**	**100.00**	**100.00**	**100.00**

Test Results

```
Contract: PopStaking
  PopStaking Deploy Phase Test Cases
    ✓ should deploy with correct token (268ms)
    ✓ should deploy with correct devaddr (375ms)
    ✓ should deploy with correct start date (213ms)
    ✓ should deploy with correct start block and claimable
      block (209ms)
    ✓ should deploy with correct pop per block cycle (504ms)
  PopStaking Deposit Phase Test Cases
    ✓ should count pop per block correctly (1421ms)
    ✓ should count claimable pop correctly (1346ms)
    ✓ should count multiplier correctly (228ms)
    ✓ should deposit pop correctly (1464ms)
    ✓ should claim pop rewards with deposit correctly (3772ms)
    ✓ should safe pop transfer work correctly (5800ms)
    ✓ shouldn't deposit to small amount (2631ms)
    ✓ should withdraw pop correctly (2086ms)
    ✓ shouldn't withdraw amount equal zero (463ms)
    ✓ shouldn't withdraw amount greater than user deposit (830ms)
    ✓ should emergency withdraw pop correctly (1135ms)
    ✓ should update pending info correctly (2371ms)
    ✓ shouldn't update pending info by not the dev (1539ms)
```

```
    ✓ shouldn't update pending info with invalid parameters (1696ms)
    ✓ should change dev address correctly (574ms)
    ✓ shouldn't change dev address by not the current dev (428ms)

  29 passing (1m)
```

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