StakingDrop Protocol

Security Assessment

December 18th, 2020

For :
StakingDrop team @ Bandot foundation ltd

By :
Zach Zhou @ CertiK
jun.zhou@certik.org

Disclaimer

CertiK Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has indeed completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.

Overview

Project Summary

Project Name	bandot-staking-drop
Description	a defi platform to distribute staking rewards
Platform	Ethereum; Solidity
Codebase	GitHub Repository
Commit	ed2eb969a1a0354a48103b9627afc6964f7c16af

Audit Summary

Delivery Date	Dec. 18, 2020
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	1
Timeline	Dec. 16, 2020 - Dec. 18, 2020

Vulnerability Summary

Total Issues	7
Total Critical	0
Total Major	0
Total Minor	1
Total Informational	6

Executive Summary

This report has been prepared for **StakingDrop** protocol to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Dynamic Analysis, Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Documentation

The sources of truth regarding the operation of the contracts in scope were lackluster and are something we advise to be enriched to aid in the legibility of the codebase as well as project. To help aid our understanding of each contract's functionality we referred to in-line comments and naming conventions.

These were considered the specification, and when discrepancies arose with the actual code behaviour, we consulted with the **StakingDrop** team or reported an issue.

Review Notes

The audited commit is ed2eb969a1a0354a48103b9627afc6964f7c16af and the file included in the scope was StakingDrop.sol.

Source Code SHA-256 Checksum

- StakingDrop.sol

f63cf102d04d35c29abc0cb64bc48094011b7e63ad7a2af3ce431d79f2d05540

Certain optimization steps that we pinpointed in the source code mostly referred to coding standards and inefficiencies, however 1 minor vulnerability was identified during our audit that solely concerns the specification.

Certain discrepancies between the expected specification and the implementation of it were identified and were relayed to the team, however they pose no type of vulnerability and concern an optional code path that was unaccounted for.

Recommendations

Overall, the codebase of the contracts should be refactored to assimilate the findings of this report, enforce linters and / or coding styles as well as correct any spelling errors and mistakes that appear throughout the code to achieve a high standard of code quality and security.

Findings

ID	Title	Type	Severity
EXH-01	Gas Consumption	Optimization	Informational
EXH-02	Redundant check	Optimization	Informational
EXH-03	Missing check for address parameters of constructor	Optimization	Informational
EXH-04	Incorrect Rewards Calculation	Optimization	Minor
EXH-05	Proper Usage of "public" and "external" type	Coding Style	Informational
EXH-06	Missing check for parameter of deposit(), withdraw()	Optimization	Informational
EXH-07	Missing check for result of transfer(),transferFrom()	Optimization	Informational

Exhibit-01: Redundant check

Type	Severity	Location
Optimization	Informational	StakingDrop.sol L46-47

Description:

There is a redundant comparison in function `withdraw()` , `block.timestamp < bonusStartAt.add(BONUS_DURATION)` include interval `block.timestamp < bonusStartAt`

```
if (block.timestamp < bonusStartAt) return;
if (block.timestamp < bonusStartAt.add(BONUS_DURATION)) return;
```

Recommendation:

We recommend to remove the first check `if (block.timestamp < bonusStartAt) return;`.

(**Bandot - Resolved**) The issue is addressed in commit 4d3ed3c59b077d9ea982ee257199f367be6481ec

Exhibit-02: Gas consumption

Type	Severity	Location
Optimization	Informational	StakingDrop.sol L14-16

Description:

Below variables change only once, better to define it as immutable to avoid gas consumption.

```
address public hbtcAddress;
address public bdtAddress;
uint256 public bonusStartAt;
```

Recommendation:

We recommend to change the codes as below:

```
address public immutable hbtcAddress;
address public immutable bdtAddress;
uint256 public immutable bonusStartAt;
```

(**Bandot - Resolved**) The issue is addressed in commit
4d3ed3c59b077d9ea982ee257199f367be6481ec

Exhibit-03: Missing check for address parameters of `constructor()`

Type	Severity	Location
Optimization	Informational	StakingDrop.sol L35-L43

Description:

Missing check for address parameters of `constructor()`.

Recommendation:

We recommend to change the codes as below:

```
constructor(
        address hbtcAddress_,
        address bdtAddress_,
        uint256 bonusStartAt_
    ) public Ownable() {
       require(hbtcAddress_ != address(0), "StakingDrop: hbtcAddress_ is zero
address");
       require(bdtAddress_ != address(0), "StakingDrop: bdtAddress_ is zero
address");
       hbtcAddress = hbtcAddress_;
       bdtAddress = bdtAddress_;
       bonusStartAt = bonusStartAt_;
    }
```

(**Bandot - Resolved**) The issue is addressed in commit
4d3ed3c59b077d9ea982ee257199f367be6481ec

Exhibit-04: Incorrect Rewards Calculation

Type	Severity	Location
Optimization	Minor	StakingDrop.sol L103-130

Description:

Function `getIncrementalRewards` may result in unfair reward distribution.

Example:

User A deposited 1 ether on the first day, no other users deposited until the end of the 7th day.

If user A claim rewards at the end of the 7th day, he will get 100% of the total rewards for all 7 days.

If user A forgot to claim rewards, and in the 8th day user B deposited 99 ethers.

At the end of the 8th day, user A claimed rewards, and he will only get down to 1% of the total

rewards for all 8 days.

Could you please explain the design of this function?

(**Bandot - Response**) This rewards formula is matched with their design.

Exhibit-05: Proper Usage of "public" and "external" type

Type	Severity	Location
Coding Style	Informational	StakingDrop.sol L45,L56

Description:

"public" functions that are never called by the contract could be declared "external".

Example:

Functions like : `withdraw()`, `deposit()`

Recommendation:

Consider using the "external" attribute for functions never called from the contract.

Example:

```
function withdraw(uint256 amount) external
```

(**Bandot - Resolved**) The issue is addressed in commit
4d3ed3c59b077d9ea982ee257199f367be6481ec

Exhibit-06: Missing check for parameter of `deposit()`, `withdraw()`

Type	Severity	Location
Optimization	Informational	StakingDrop.sol L45,L56

Description:

`_value` and `amount` should greater than 0.

Recommendation:

Consider adding require check for parameter `amount`, `_value` like below:

```
require(amount > 0, "StakingDrop: amount should greater than zero");
require(_value > 0, "StakingDrop: _value should greater than zero");
```

(**Bandot - Resolved**) The issue is addressed in commit
4d3ed3c59b077d9ea982ee257199f367be6481ec

Exhibit-07: Missing check for result of `transfer()`, `transferFrom()`

Type	Severity	Location
Optimization	Informational	StakingDrop.sol L52,L64,L82

Description:

Values of `myDeposit`, `totalDeposit` had been changed before `transfer()`, `transferFrom()`

If `transfer()`, `transferFrom()` return false(means transfer failed), values of `myDeposit`, `totalDeposit` will be incorrect since transfer unsuccessfully, all the value change should be reverted .

Recommendation:

Consider using `safeTransfer()`, `safeTransferFrom()` provided by `SafeERC20` to replace `transfer()`, `transferFrom()` or using `require` validation to check transfer result.

Example:

```
require(IERC20(hbtcAddress).transfer(msg.sender, amount), "StakingDrop:
withdraw transfer failed");
```

(**Bandot - Resolved**) The issue is addressed in commit 4d3ed3c59b077d9ea982ee257199f367be6481ec, 080088c368df9926f920206e860f7f8b7a32e25b