CERTIK

Popsicle Finance

Core Contracts

Security Assessment

May 5th, 2021

Disclaimer

CertiK reports are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security review.

CertiK Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

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 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	Popsicle Finance - Core Contracts
Description	A SushiSwap based fork of the full staking and token system.
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commits	1. 54be2ce3cf53738b24f3518575d0ce3e2f209c09 2. c968adde157b0cc929cef11de4500caf0ef4881a

Audit Summary

Delivery Date	May 5th, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	1
Timeline	April 8th, 2021 - April 10th, 2021

Vulnerability Summary

Total Issues	8
● **Total Critical**	0
● **Total Major**	1
● **Total Medium**	1
● **Total Minor**	3
● **Total Informational**	3

Executive Summary

We were tasked with auditing the codebase of Popsicle Finance and namely, their staking reward mechanisms based on SushiSwap.

Over the course of the audit we were able to identify three important findings that we believe should be remediated as soon as possible to consider the codebase in a deployable state and relate to the proper functionality of vesting and staking mechanisms.

The codebase contains an adjusted `MasterChef` implementation of SushiSwap that rewards pools on a per-second reward rate instead of a per-block reward rate and otherwise operates similarly to the original implementation. The documentation of the project should be improved as no `README` accompanied the code and its functionality was mostly deduced by the code itself as well as any comments that were introduced to it.

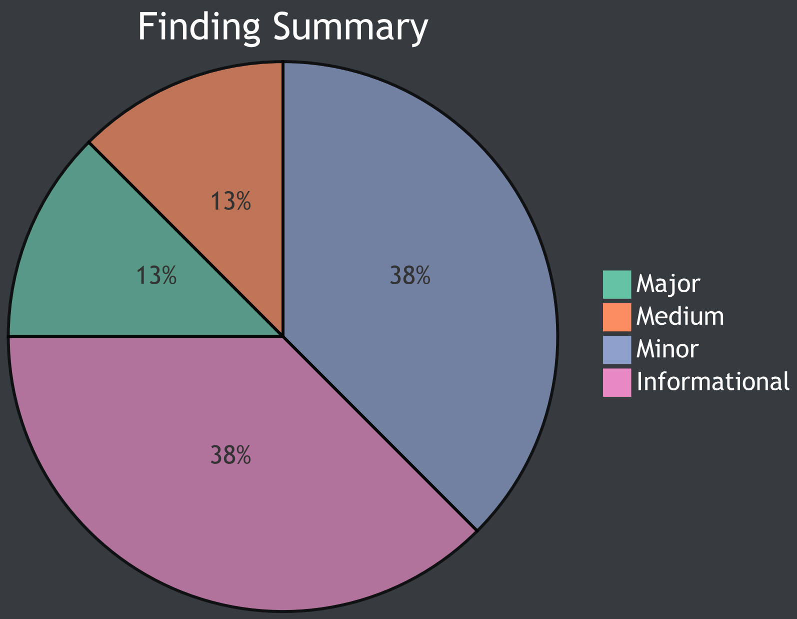
System Analysis

The `owner` of the `IceToken` is able to arbitrarily mint and burn tokens from and to addresses respectively. As the Popsicle team has stated that they intentionally dropped the minting functionality from their `MasterChef` implementation, we believe the owner to be an EOA controlled by the Popsicle team and as such we advise due diligence to be applied by both the Popsicle team and its users as compromisation of the private keys can have devastating consequences to the overall protocol.

Files In Scope

ID	Contract	Location
DHS	DiamondHands.sol	DiamondHands.sol
ITN	IceToken.sol	IceToken.sol
PJT	PopsicleJoint.sol	PopsicleJoint.sol
PPV	PopsicleProjectVesting.sol	PopsicleProjectVesting.sol
PSD	PopsicleStand.sol	PopsicleStand.sol
SOR	Sorbettiere.sol	Sorbettiere.sol

File Dependency Graph



Manual Review Findings

ID	Title	Type	Severity	Resolved
DHS-01	Pull-Over-Push Pattern	Logical Issue	● Minor	✓
DHS-02	Inexistence of Checks-Effects-Pattern	Logical Issue	● Minor	✓
PJT-01	Contract Freeze	Logical Issue	● Major	✓
PPV-01	Circumvention of Vesting	Logical Issue	● Minor	✓
PPV-02	Strict Conditional	Coding Style	● Informational	✓
PSD-01	Suboptimal Deletion of Storage	Coding Style	● Informational	✓
SOR-01	Incorrect Withdrawal of Funds	Logical Issue	● Medium	✓
SOR-02	Suboptimal Deletion of Storage	Coding Style	● Informational	ⓘ

DHS-01: Pull-Over-Push Pattern

Type	Severity	Location
Logical Issue	● Minor	DiamondHands.sol L84-L88

Description:

The `transferOwnership` function overrides the current `_owner` with the `newOwner` without ensuring that the `newOwner` is able to actuate transactions on the blockchain.

Recommendation:

We advise the pull-over-push pattern to be applied whereby a new owner is proposed and needs to consequently accept ownership via a dedicated function ensuring that they are able to transact with the contract and are aware of the ownership. This finding applies to all `Ownable` implementations in the flattened contracts but will not be repeated for the sake of brevity.

Alleviation:

A new pattern was used whereby the `transferOwnership` function accepts two `bool` variables that indicate how it should behave i.e. whether it should directly overwrite the previous owner or assign them to the `pendingOwner` slot and they consequently need to accept ownership.

We should note that we believe the `pendingOwner` should be reset when a `direct` transfer of ownership is utilized to prevent misbehaviours from arising.

DHS-02: Inexistence of Checks-Effects-Pattern

Type	Severity	Location
Logical Issue	● Minor	DiamondHands.sol L507-L511

Description:

The `withdraw` function performs a transfer of rewards without incrementing the `withdrawedAmount` member of the `user` struct beforehand.

Recommendation:

We advise that the `user.withdrawedAmount` variable is incremented prior to the external call either within `safeRewardTransfer` or in the linked code block to ensure the code conforms to the Checks-Effects-Interactions pattern properly.

Alleviation:

The `safeRewardTransfer` function was reworked to instead just return the amount to be transferred (now called `getSafeRewardTransferAmount`) and the code block that invoked this function now properly increments the user's `withdrawedAmount` before performing the `transfer` of the token.

PJT-01: Contract Freeze

Type	Severity	Location
Logical Issue	● Major	PopsicleJoint.sol L587

Description:

The `safeApprove` function of OpenZeppelin does not perform as expected and will cause the contract to freeze on the second `stake` being made as the `safeApprove` function internally asserts that the address being approved has a zero approval when set to a non-zero approval.

Recommendation:

We advise that either the approval is zeroed out before this call or that the `safeApprove` function is dropped entirely in favor of `approve`, the former of which we advise as the `safeApprove` wrapper conducts the opportunistic evaluation of the return value.

Alleviation:

The `safeApprove` invocation was replaced by a direct `approve` invocation which should be considered safe in the case of most LP token implementations. We still advise the Popsicle team to apply caution when introducing new LP tokens to the system ensuring that they are fully supported.

PPV-01: Circumvention of Vesting

Type	Severity	Location
Logical Issue	● Minor	PopsicleProjectVesting.sol L534-L537

Description:

The `retrieveExcessTokens` enables the `owner` to preemptively acquire the vested tokens by simply transferring them outwards at any time.

Recommendation:

We advise the function to solely be invoke-able after the 156th week as that is the intended purpose judging by its naming implying "excess" tokens.

Alleviation:

A new `require` check was introduced ensuring that the `block.timestamp` has surpassed the `_releaseTime` and thus preventing early redemption of the tokens.

PPV-02: Strict Conditional

PPV-02: Strict Conditional

Type	Severity	Location
Coding Style	● Informational	PopsicleProjectVesting.sol L489, L519

Description:

The `vestingAmount` that is meant to be returned beyond week `156` as the comment of L450 indicates is equal to `FOR_156_WEEK`, however, it is solely returned for the 156th week and beyond that no rewards are given.

Recommendation:

We advise the adjustment of either the comment or the conditional to conform to the desired purpose.

Alleviation:

The comment was properly adjusted to reflect the variable's functionality.

PSD-01: Suboptimal Deletion of St

PSD-01: Suboptimal Deletion of Storage

Type	Severity	Location
Coding Style	● Informational	PopsicleStand.sol L682-L686

Description:

The linked assignments manually zero out all members of the `UserInfo` struct.

Recommendation:

We advise the `delete` operation to be utilized instead to ensure that even an update to the members of the `UserInfo` struct does not break this functionality.

Alleviation:

The `delete` operation is now properly utilized in the `emergencyWithdraw` function.

SOP-01: Incorrect Withdrawal of Funds

SOR-01: Incorrect Withdrawal of Funds

Type	Severity	Location
Logical Issue	● Medium	Sorbettiere.sol L646-L655

Description:

The funds withdrawn during an emergency are not accounted for in the `stakingTokenTotalAmount` causing future rewards to be diluted incorrectly.

Recommendation:

We advise the `stakingTokenTotalAmount` member to be updated properly on the `pool` to ensure no such issue arises.

Alleviation:

The `stakingTokenTotalAmount` is now properly updated whenever an `emergencyWithdraw` is performed. Additionally, the code was updated to use the `delete` operation recommended in another file's findings properly.

SOR-02: Suboptimal Deletion of Storage

SOK-02: Suboptimal Deletion of Storage

Type	Severity	Location
Coding Style	● Informational	Sorbettiere.sol L650-L652

Description:

The linked assignments manually zero out all members of the `UserInfo` struct.

Recommendation:

We advise the `delete` operation to be utilized instead to ensure that even an update to the members of the `UserInfo` struct does not break this functionality.

Alleviation:

The Popsicle Finance - Core Contracts development team has not provided a response to this exhibit yet.

Appendix

Appendix

Finding Categories

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.