CERTIK

1Hive

HoneySwap Contracts

Security Assessment

April 22nd, 2021

Audited By:

Alex Papageorgiou @ CertiK

alex.papageorgiou@certik.org

Reviewed By:

Camden Smallwood @ CertiK

camden.smallwood@certik.org

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	1Hive - HoneySwap Contracts
Description	A SushiSwap-based farming implementation
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commits	1. 1dcebf24dd7350ea8c3dced861ac28a6122e143a 2. 6c8c6d076e15aec0c0f04a24a564fd33c4d7984e

Audit Summary

Delivery Date	April 22nd, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	1
Timeline	April 6th, 2021 - April 7th, 2021

Vulnerability Summary

Total Issues	6
🔴 **Total Critical**	0
🟠 **Total Major**	0
🟡 **Total Medium**	1
🔵 **Total Minor**	3
🟢 **Total Informational**	2

Executive Summary

We were tasked with auditing the HoneyFarm codebase of 1Hive, a SushiSwap based implementation that incorporates the ERC721 concept to deposits and permits them to be freely transacted while they continue accruing rewards.

The codebase of the project has been developed conforming to the latest style guideliens and sufficient documentation as well as legible function and variable names have been defined greatly enhancing the reading experience of the codebase.

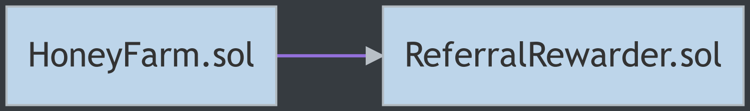
We identified an inaccuracy in the calculation performed for an event within `ReferralRewarder` as well as a potential issue with the way locked rewards are calculated that allow a user to circumvent the `maxTimeLock` restriction and continue accruing locked multiplier rewards.

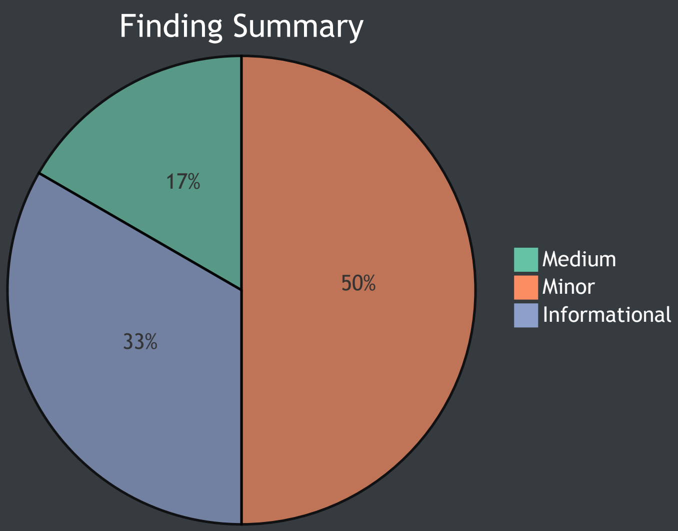
We advise the 1Hive team to promptly remediate the issues outlined above to ensure the codebase is of a high security standard.

Files In Scope

ID	Contract	Location
HSF	HSFToken.sol	contracts/HSFToken.sol
HFM	HoneyFarm.sol	contracts/HoneyFarm.sol
RRR	ReferralRewarder.sol	contracts/ReferralRewarder.sol

File Dependency Graph





Manual Review Findings

ID	Title	Type	Severity	Resolved
HFM-01M	Incorrect Lock Reward Calculation	Logical Issue	● Medium	✓
HFM-02M	Formula Inaccuracy	Mathematical Operations	● Minor	✓
HFM-03M	Insufficient Sanitization of Unlock Time	Logical Issue	● Minor	✓
HFM-04M	Inexistent Deletion of Deposits	Logical Issue	● Informational	✓
RRR-01M	Incorrect Missing Reward Calculation	Mathematical Operations	● Minor	✓

Static Analysis Findings

ID	Title	Type	Severity	Resolved
HFM-01S	Event Emitted Out of Order	Language Specific	● Informational	✓

HFM-01M: Incorrect Lock Reward Calculation

Type	Severity	Location
Logical Issue	● Medium	HonevFarm.sol L372-L380

Description:

The `_downgradeExpired` function calculates the reward of a locked deposit until the current `block.timestamp` rather than until the time the lock expires, causing incorrect rewards to be calculated and allowing users to lock their deposits as long as they want to continueing to accrue the rewarded multiplier of the locked deposit.

Recommendation:

We advise the function to factor the `unlockTime` of a `deposit` properly to ensure calculations are done fairly and lock rewards are distributed properly.

Alleviation:

The Hive team has stated that this is a known issue due to the constraints of the system itself as it is hard to accurately track the rewards a particular deposit would amass until the exact `unlockTime` and retro-active application of rewards is also challenging. As a mitigation action, the Hive team has incentivized the invocation of `downgradeExpired` function by introducing a `downgradeReward` that is sent outward during a `_downgradeExpired` invocation, partially alleviating this issue.

HFM-02M: Formula Inaccuracy

Type	Severity	Location
Mathematical Operations	● Minor	HoneyFarm.sol L219, L221

Description:

The comments above the implemented formula within `getDistribution` denote that the calculation should reflect `(t2 - t1) * (2 * ds - (-m) * (t2 + t1)) / 2` whereas it reflects `(t2 - t1) * (2 * ds - (m * (t2 + t1))) / 2`.

Recommendation:

We advise the correct formula to either be reflected in the comment or implemented by the contract.

Alleviation:

After discussion with the Hive team, we have concluded that the implemented formula is indeed the correct one as the `distributionSlope` variable represents `-m` thereby nullifying the signs of the formulas. As a result, this exhibit is nullified. We still advise that the Hive team properly documents this "discrepancy" in the code to aid future readers.

HFM-03M: Insufficient Sanitization of Unlock Time

Type	Severity	Location
Logical Issue	● Minor	HoneyFarm.sol L228, L252, L254, L259

Description:

In the current system, if the `timeLockConstant` is greater-than (`>`) the `SCALE` variable, it is possible to acquire this benefit while specifying a lock time of 1 second as there is no minimum lock time specified.

Recommendation:

We advise a minimum lock time to be specified to ensure users cannot circumvent the lock system to acquire the timelock bonus without practically locking their deposit.

Alleviation:

A minimum lock time was introduced into the system thereby alleviating this issue.

HFM-04M: Inexistent Deletion of Deposits

Type	Severity	Location
Logical Issue	● Informational	HoneyFarm.sol L283-L302

Description:

While a deposit cannot be closed again due to the usage of an ERC721 to represent it that is burned, the data still resides on the blockchain and cannot be easily differentiated from valid data.

Recommendation:

We advise the `deposit` to be properly deleted from the blockchain once the `closeDeposit` call concludes to ensure the contract does not become polluted with useless information.

Alleviation:

Deposits are properly deleted as the last statement within `closeDeposit` addressing this issue.

RRR-01M: Incorrect Missing Reward Calculation

Type	Severity	Location
Mathematical Operations	● Minor	ReferralRewarder.sol L31

Description:

The `MissingReward` emitted during `distributeReward` accepts an `owedReward` argument indicating how much reward is pending for a user to potentially be used by an off-chain process to re-invoke `distributeReward` at a later date.

Recommendation:

The calculation is incorrect because the subtraction `currentReserves - reward` will always underflow causing an enormous award being emitted and consequently exploited by a user due to the off-chain processes caching this value for future rewards. We advise the calculation to be corrected by inversing the order (i.e. `reward - currentReserves`) to properly `emit` the amount missing from the reward payout.

Alleviation:

The proper calculation was introduced in the `MissingReward` event emittance of the `distributeReward` function thereby alleviating this exhibit.

HFM-01S: Event Emitted Out of Order

Type	Severity	Location
Language Specific	● Informational	HoneyFarm.sol L277-L279

Description:

The `Referred` event may be emitted out of order if a user re-enters the contract due to the ERC721 callback that occurs within the `_safeMint` function.

Recommendation:

We advise the `if` clause and accompanying event to be relocated before the `_safeMint` call ensuring orderly execution of code.

Alleviation:

The function was re-ordered to emit the event before the mint occurs, thereby ensuring it is minted in order.

Appendix

Finding Categories

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

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.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of `private` or `delete`.