HashEx

Yield Farming
Smart Contracts Security Audit

Prepared for: Yield Farming Team

September 17, 2020

By:

HashEx
https://hashex.org

Contents

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and HashEx and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (HashEx) owe no duty of care towards you or any other person, nor does HashEx make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and HashEx hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, HashEx hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against HashEx, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

Background

HashEx was commissioned by the Yield Farming team to perform an audit of YieldToken, YieldChef, Timelock and TimelockWrapper smart contracts. The audit was conducted between September 9 and September 17, 2020.

The contracts were deployed at 0xf7cb015141feb3814acd5f685fbfe879326bcadc (YieldChef), 0x6768063279e2b185dc0c972b97f11f231d0b45ad (YieldToken), 0x56033e81ab1179b6cb29b8e4a7a1065a0ae3ec8b (Timelock), 0x511a878bd440f15c0f6c86590a9a6f3c58d9284b (TimelockWrapper).

The purpose of this audit was to achieve the following:

- Ensure that smart contracts function as intended.
- Identify potential security issues with smart contracts.

Information in this report should be used to understand the risk exposure of smart contracts, and as a guide to improve the security posture of smart contracts by remediating the issues that were identified.

YieldToken smart contract

No critical, high or medium severity issues were found. Contract was implemented with a well known OpenZeppelin library.

Low Severity Issues and General Recommendations

1. The compiler version should be fixed to avoid any potential discrepancies in smart contract behavior caused by different compiler versions. Line 11.

YieldChef smart contract

Medium Severity Issues

1. `function add` has no check to prevent the same LP token to be added more than once. In such a case variables associated with the pool of the token will be overridden which will mess up distribution of rewards to the pools and the stakers. We recommend adding an additional `require` sanity check in the smart contract or at least programmatically perform this check offchain. Line 977.
2. Possible reentrancy attack at `function emergencyWithdraw` and `function deposit`. This attack is possible if LP token implements code for this attack. We recommend rewriting the code using checks-effects-interactions pattern or performing audits of newly added tokens. Lines 1000, 1029.
3. Helper `function getMultiplier` can return a wrong result if called with parameter `_from` less than `startBlock`. This function is also used in `function`

`pendingYield` and `function updatePool` however these functions always call with `_from` param less that `startBlock`. Possible fix is to set `_from` to `startBlock` if `_from` is less that `startBlock`. Line 943.

4. Comment in the `function updatePool` states the amount of the reward to the `devaddr` is 7.5%. Actually it is slightly less than 0.7%. Line 993.

Low Severity Issues and General Recommendations

1. Calling `function massUpdatePools` may result in using more gas than block gas limit. It also affects `function add` and `function set`. However it is possible to call functions without usage of `function massUpdatePools`. Line 970.
2. `function dev` to change the `devaddr` address cannot be called. The YieldChef contract was deployed with the `devaddr` address set to 0x35c8c0d515cb451c41295bda469f96b1ac98b6ae. At this address is deployed `AdminAccount` smart contract and it has no functionality to call YieldChef's `function dev`. Line 1049.
3. Typos in comments. Words "multiplier", "points" are misspelled. Lines 881, 888.
4. Comments use the forked contract name (`MasterChef` instead of `YieldChef`). Lines 999, 1015.
5. `variable BONUS_MULTIPLIER` is redundant. Line 882.
6. `function mint` cannot be called as the `YieldChef` contract if owned by the `TokenTimelock` contract and the admin of `TokenTimelock` is the `TimelockWrapper` contract that restricts calling other functions than function set or function add on `YieldChef` smart contract. Line 977.
7. The compiler version should be fixed to avoid any potential discrepancies in smart contract behavior caused by different compiler versions. Line 11.

Timelock smart contract

No critical, high or medium severity issues were found.

Low Severity Issues and General Recommendations

1. The compiler version should be fixed to avoid any potential discrepancies in smart contract behavior caused by different compiler versions. Line 9.
2. To favor explicitness, all instances of uint should be declared as `uint256`. Lines 155-162, 166, 172, 184, 214, 234.
3. `function setDelay` cannot be called. It can be called only by the admin of the `Timelock` contract, but the admin is set to the `TokenWrapper` contract and this contract has no functionality to call described function or to transfer admin privileges to another contract or account. Line 184.

TimelockWrapper smart contract

The `TimelockWrapper` is a proxy contract to restrict calling functions in the `YieldChef` smart contract to only `function add` and `functions set`. The `Timelock` contract is the owner of `YieldChef` contract. `TimelockWrapper` is the admin of the `Timelock` contract.
No critical, high or medium severity issues were found.

Low Severity Issues and General Recommendations

1. The compiler version should be fixed to avoid any potential discrepancies in smart contract behavior caused by different compiler versions. Line 5.
2. Duplicated `pragma` identifier. Line 151.
3. To favor explicitness, all instances of uint should be declared as `uint256`. Lines 346, 350, 354.
4. Consider adding `require` message to provide a comprehensive error description. Line 342.

Conclusion

No serious issues were found with the `YieldToken`, `Timelock` or `TimelockWrapper` smart contracts.
`YieldChef` is a fork of Sushi's MasterChef contract. It has no migration functionality, has additional `function mint` (that according to deploying setup cannot be called by anyone) and dev reward amount is set to about 7%.
Several medium severity issues were found in the `YieldChef` smart contract.
Audit includes recommendations on the code improving and preventing potential attacks.