#HashEx
BLOCKCHAIN SECURITY

Apeswap Banana Maximizer

smart contracts final audit report

April 2022

hashex.org

contact@hashex.org

Contents

1. Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice 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 disclaimer below – please make sure to read it in full.

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 the 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. HashEx owns all copyright rights to the text, images, photographs, and other content provided in the following document. When using or sharing partly or in full, third parties must provide a direct link to the original document mentioning the author (hashex.org).

2. Overview

HashEx was commissioned by the **ApeSwap** team to perform an audit of their smart contract. The audit was conducted between **05/04/2022** and **08/04/2022**.

The purpose of this audit was to achieve the following:

- Identify potential security issues with smart contracts
- Formally check the logic behind given smart contracts.

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

The code is available at the @ApeSwapFinance/apeswap-vaults GitHub repository and was audited after the commit 5b12d28.

2.1 Summary

Project name	Apeswap Banana Maximizer
URL	https://apeswap.finance
Platform	Binance Smart Chain
Language	Solidity

2.2 Contracts

Name	Address
BananaVault	

MaximizerVaultApe	
BaseBananaMaximizerStrategy	
StrategyMaximizerMasterApe	
KeeperMaximizerVaultApe	

3. Found issues

Medium	4 (33%)
Low	6 (50%)
Info	2 (17%)

C1. BananaVault

ID	Severity	Title	Status
C1-01	Low	Gas optimization	Open

C2. MaximizerVaultApe

ID	Severity	Title	Status
C2-01	Medium	Unlimited withdrawFeePeriod period	Open
C2-02	Low	Loop handling	Open
C2-03	Low	Gas optimization	Open
C2-04	Low	Unused functionality	Open
C2-05	Info	Lack of documentation	Open

C3. BaseBananaMaximizerStrategy

ID	Severity	Title	Status
C3-01	Medium	Possibility of 100% reward fees	Open
C3-02	Medium	Unlimited withdrawFeePeriod period	Open
C3-03	Low	No checks for constructor paramaters	Open

C4. StrategyMaximizerMasterApe

ID	Severity	Title	Status
C4-01	Medium	Lack of the emergency withdraw functionality	Open
C4-02	Info	Typo in documentation	Open

C5. KeeperMaximizerVaultApe

ID	Severity	Title	Status
C5-01	Low	Gas optimization	Open

4. Contracts

C1. BananaVault

Overview

The contract allows staking Banana tokens to the MasterApe contract by MaximizerVaultApe contract or by StrategyMaximizerMasterApe contract to get rewards.

Issues

C1-01 Gas optimization

Low Open

The variables `lastDepositedTime`, `lastUserActionTime` of the `UserInfo` structure can be packed together into one slot by using an integer type of `uint128`.

C2. MaximizerVaultApe

Overview

The contract allows users to stake tokens in the different vaults (strategies).

Users get rewards in banana tokens for their stake.

Issues

C2-01 Unlimited withdrawFeePeriod period

Medium Open

The contract owner has the ability to set any `withdrawFeePeriod` using the `setWithdrawFeePeriod()` function. This will cause the users to pay the fee of their rewards every time.

C2-02 Loop handling

Low Open

Due to the use of the `continue` keyword L137, the loop will continue to run and spend gas until it has been completed, because the `actualLength` variable will never change again. But instead, the `break` keyword can be used.

C2-03 Gas optimization

Low Open

a. Since the arguments `_pids` (L224), `_vaults` (L508) of the functions `earnSome()`, `addVaults()` are read-only, they can be declared as `calldata` instead of `memory` to save gas.

b. The functions `getSettings()`, `addVaults()`, `setModerator()`, `setMaxDelay()`, `setMinKeeperFee()`, `setSlippageFactor()`, `setMaxVaults()` can be declared as external to save gas.

c. Since the contract is inherited from the Sweeper contract the function `inCaseTokensGetStuck()` looks redundant and can be deleted.

d. The internal function `_approveTokenIfNeeded()` is never used in the contract and can be deleted to save gas on deployment.

e. The state variable `BANANA_VAULT` L60 can be declared as `immutable` to save gas.

C2-04 Unused functionality

Low Open

The state variable `moderator` is never used in the contract code, except the `setModerator()` function.

Recommendation

Check about its necessity in the contract.

C2-05 Lack of documentation

Info Open

The function `balanceOf()` hasn't NatSpec documentation.

Recommendation

We recommend adding NatSpec documentation to all `public` and `external` functions.

C3. BaseBananaMaximizerStrategy

Overview

The abstract contract of the farm strategy implements full functionality to interact with the token farms.

Issues

C3-01 Possibility of 100% reward fees

Medium Open

The contract owner has the ability to set the rewards fee of 100% using the `setWithdrawRewardsFee()` function. This may result in users not receiving any rewards.

Recommendation

We recommend capping reward fees.

C3-02 Unlimited withdrawFeePeriod period

Medium Open

The contract owner has the ability to set any `withdrawFeePeriod` using the `setWithdrawFeePeriod()` function. This will cause the users to pay the fee of their rewards every time.

Recommendation

We recommend capping withdrawal periods.

C3-03 No checks for constructor paramaters

Low Open

The constructor parameters are not all checked and cannot be updated later. This can lead to errors in the use of the contract if it was initialized with the wrong values.

Recommendation

We recommend using a non-zero address check.

C4. StrategyMaximizerMasterApe

Overview

The contract implements BaseBananaMaximizerStrategy contract with defined token farms.

Issues

C4-01 Lack of the emergency withdraw functionality

Medium Open

The contract has no emergency function to perform the `emergencyWithraw()` function of any farm. So the users' funds can be blocked in the farm, because of breaking rewards functionality.

Recommendation

We strongly recommend adding the 'emergency withdraw' functionality (withdrawal without rewards) at least callable by the contract owner.

C4-02 Typo in documentation

Info Open

The parameter `_amount` if the function `_vaultDeposit()` has the incorrect description (the 'remove' word).

C5. KeeperMaximizerVaultApe

Overview

The contract is used to call the function of the MaximizerVaultApe contract using ChainLink Keepers.

Issues

C5-01 Gas optimization

Low Open

The function `setKeeper()` can be declared as `external` to save gas.

5. Conclusion

4 medium, 6 low and 2 informational severity issues were found.

This audit includes recommendations on improving the code and preventing potential attacks.

Appendix A. Issues severity classification

- **Critical.** Issues that may cause an unlimited loss of funds or entirely break the contract workflow. Malicious code (including malicious modification of libraries) is also treated as a critical severity issue. These issues must be fixed before deployments or fixed in already running projects as soon as possible.
- **High.** Issues that may lead to a limited loss of funds, break interaction with users, or other contracts under specific conditions. Also, issues in a smart contract, that allow a privileged account the ability to steal or block other users' funds.
- **Medium.** Issues that do not lead to a loss of funds directly, but break the contract logic. May lead to failures in contracts operation.
- **Low.** Issues that are of a non-optimal code character, for instance, gas optimization tips, unused variables, errors in messages.
- **Info.** Issues that do not impact the contract operation. Usually, info severity issues are related to code best practices, e.g. style guide.

Appendix B. List of examined issue types

- Business logic overview
- Functionality checks
- Following best practices
- Access control and authorization
- Reentrancy attacks
- Front-run attacks
- DoS with (unexpected) revert
- DoS with block gas limit
- Transaction-ordering dependence
- ERC/BEP and other standards violation
- Unchecked math
- Implicit visibility levels
- Excessive gas usage
- Timestamp dependence
- Forcibly sending ether to a contract
- Weak sources of randomness
- Shadowing state variables
- Usage of deprecated code

 contact@hashex.org

 @hashex_manager

 blog.hashex.org

 linkedin

 github

 twitter

#HashEx
BLOCKCHAIN SECURITY