AUDITS

SECURITY ASSESSMENT

PANCAKESWAP

APRIL 27TH 2022

TABLE OF CONTENTS

1 LEGAL DISCLAIMER

2 MH AUDITS INTRO

3 PROJECT SUMMARY

4 AUDIT SCORES

5 AUDIT SCOPE

6 KEY FINDINGS

7 VULNERABILITIES

8 SOURCE CODE

9 APPENDIX

LEGAL DISCLAIMER

MH Audits 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 MH Audits to perform a security review.

MH Audits does 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.

MH Audits 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.

MH Audits represents 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. MH Audits’ position is that each company and individual are responsible for their own due diligence and continuous security.

MH Audits’ 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.

INTRODUCTION

MH Audits is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

Secure your project with MH Audits

We offer field-proven audits with in-depth reporting and a range of suggestions to improve and avoid contract vulnerabilities.

Industry-leading comprehensive and transparent smart contract auditing on all public and private blockchains.

Vunerability checking

A crucial manual inspection carried out to eliminate any code flaws and security loopholes. This is vital to avoid vulnerabilities and exposures incurring costly errors at a later stage.

Contract verification

A thorough and comprehensive review in order to verify the safety of a smart contract and ensure it is ready for launch and built to protect the end-user.

Risk assessment

Analyse the architecture of the blockchain system to evaluate, assess and eliminate probable security breaches. This includes a full assessment of risk and a list of expert suggestions.

In-depth reporting

A truly custom exhaustive report that is transparent and depicts details of any identified threats and vulnerabilities and classifies those by severity.

Fast turnaround

We know that your time is valuable and therefore provide you with the fastest turnaround times in the industry to ensure that both your project and community are at ease.

Best-of-class blockchain engineers

Our engineers combine both experience and knowledge stemming from a large pool of developers at our disposal. We work with some of the brightest minds that have audited countless smart contracts over the last 4 years.

PROJECT SUMMARY

This report has been prepared for PancakeSwap to discover issues and vulnerabilities in the source code ofthe PancakeSwap project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing 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 producedby industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommendaddressing these findings to ensure a high level of security standards and industry practices. We suggestrecommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified inpublic;
- Provide more transparency on privileged activities once the protocol is live.

Project Name *PancakeSwap*

Platform *Ethereum*

Language *Solidity*

Codebase *https://github.com/pancakeswap/pancake-farm*

Commit *05e7fbdd16a94b7e18b51811eda411fb4a1b4b41*
daf8da084170ec8fe7ff705a7c0489dc8f730e50

AUDIT SCORES

Issues	9
Critical	0
Major	1
Medium	1
Minor	1
Informational	6
Discussion	0

All issues are described in further detail on the following pages.

AUDIT SCOPE

FILE	SHA256 CHECKSUM
MasterChef.sol	61cda494749c052bc0abc18e9eee5d365924112c9b028a15b-328d3615231183e
SousChef.sol	b0479b49ee6a067d6df573efc58b260b0b507ae033bf9ea77dc-cfad9316eed61
SousChef.sol	d3503231fb16984ce783991c27cd1d8270b6615677f7b002ba3a284911f5c48c

KEY FINDINGS

TITLE	SEVERITY	STATUS
Variable Naming Convention	Informational	*Acknowledged*
Comment Typo	Informational	*Resolved*
Centralized Control of Bonus Multiplier	Informational	*Resolved*
Assignment Optimization	Informational	*Resolved*
Incorrect Delegation Flow	Major	*Resolved*
Inexistent Delegate Transfer	Medium	*Resolved*
addressList Inaccuracy	Minor	*Resolved*
Contract Purpose Unclear	Informational	*Resolved*
Incorrect Reset Mechanism	Informational	*Resolved*

Description:

The linked variables do not conform to the standard naming convention of Solidity whereby functions andvariable names utilize the camelCase *format unless variables are declared as* constant *in which case theyutilize the* UPPER_CASE *format*

Location: *MasterChef.sol: 71*

Issue: *Variable Naming Convention*

Level: *Informational*

Recommendation: *We advise that the naming conventions utilized by the linked statements are adjusted to reflect the correct type of declaration according to the Solidity style guide.*

Alleviation: *The PancakeSwap development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase due to time constraints.*

Description:

The linked comment statement contains a typo in its body, namely poitns*.*

Location: *MasterChef.sol: 79*

Issue: *Comment Typo*

Level: *Informational*

Recommendation: *We advise that the comment text is corrected.*

Alleviation: *The comment typo was properly fixed.*

Description:

The function updateMultiplier *can alter the* BONUS_MULTIPLIER *variable and consequently the output of* getMultiplier *which is directly utilized for the minting of new cake tokens.*

Location: *MasterChef.sol: 113~115*

Issue: *Centralized Control of Bonus Multiplier*

Level: *Informational*

Recommendation: *This is intended functionality of the protocol, however users should be aware of this functionality.*

Alleviation: *The PancakeSwap team informed us that there is a 6-hour timelock on the MasterChef contract with regards to all pool reward changes which are also first voted on by SYRUP holders through their voting portal using a Snapshot mechanism. This decentralizes the aspect of changing the multipliers via governance by SYRUP holders.*

Description:

TThe linked statement will only yield a different output stored to totalAllocPoint *only if the condition of L146 yields true*

Location: *MasterChef.sol: 143*

Issue: *Assignment Optimization*

Level: *Informational*

Recommendation: *As a result of the above, it is more optimal to move the assignment of L143 to the* if *block of L146.*

Alleviation: *The assignment was properly moved to the linked if block, optimizing the code segment.*

Description:

Whenever new SYRUP tokens are minted, new delegates are moved from the zero address to the recipientof the minting process. However, whenever tokens are burned, new delegates are once again moved fromthe zero address to the recipient whereas delegates should be moved on the opposite way.

Location: *SyrupBar.sol: 17*

Issue: *Incorrect Delegation Flow*

Level: *Major*

Recommendation: *We advise that the* address(0***) and*** _from *variable orders are swapped on L17 to alleviate this issue. At its current state, it breaks the delegate mechanism and can also lead to a user being unable to mint / burn tokens in case the upper limit of a* uint256 *is reached due to the* SafeMath ***utilization on L233.***

Alleviation: *The delegation flow was fixed in the source code of the GitHub repository, however, the issue still persists in the deployed version of PancakeSwap. However, the* SYRUP *token will not be utilized for the DAO governance by the PancakeSwap team.*

Description:

The transfer *and* transferFrom *functions of the YAM project transfer delegates as well via overridence. The PancakeSwap implementation does not, leading to an inconsistency in the delegates of each* address*.*

Location: *SyrupBar.sol: 1*

Issue: *Inexistent Delegate Transfer*

Level: *Medium*

Recommendation: *We advise that the* transfer *and* transferFrom *functions are properly overriden to also transfer delegates on each invocation from the sender of the funds to the recipient.*

Alleviation: *TAfter evaluating with PancakeSwap, we came to the conclusion that this functionality is unnecessary as delegates are not and will not be utilized in any form of DAO governance mechanism.*

Description:

The first linked if *block pushes a new* address *to the* addressList *array in the case the* userInfo *mapping lookup yields* 0 *on the* amount *member. This case is possible even after the user has already been added to the array, either by invoking* emergencyWithdraw *or withdrawing the full amount held by the user.*

Location: *SousChef.sol: 120~122*

Issue: addressList ***Inaccuracy***

Level: *Minor*

Recommendation: *We advise that the* push *mechanism is revised to ensure that the user does not already exist in the array.*

Alleviation: *The PancakeSwap team altered the condition for pushing new items to the* addressList *array, however, duplicates can still exist. After conversing with the team, we were informed that the array is not utilized on-chain and is meant to aid off-chain processes in an airdrop mechanism that will eliminate duplicate addresses. As such, this issue can be safely ignored. We would like to note that this is not an optimal mechanism to conduct this, as it would be better to instead rely on emitted* event *s and blockchain analysis rather than contract storage.*

Description:

The SousChef *contract tracks a reward schedule based on the deposited SYRUP tokens, however, the variables of the* UserInfo *struct are never actually utilized to provide any reward.*

Location: *SousChef.sol: 1~156*

Issue: *Contract Purpose Unclear*

Level: *Informational*

Recommendation: *We advise that further documentation is produced that details the purpose of the contract, as it should seemingly interoperate with another contract that reads data from it.*

Alleviation: *The purpose of the contract is for SYRUP holders to stake their tokens and accumulate rewards on paper rather than on-chain which will be then distributed by the PancakeSwap team. As such, we believe that the purpose of the contract has been sufficiently described. We would like to note that this type of distribution of rewards purely relies on the honesty of PancakeSwap and does not utilize any on-chain or decentralized mechanisms.*

Description:

The emergencyWithdraw *function is meant to "reset" a user's state and withdraw his deposited tokens. In this case, the* rewardPending *variable of the* user *struct is not zeroed out.*

Location: *SousChef.sol: 148~154*

Issue: *Incorrect Reset Mechanism*

Level: *Informational*

Recommendation: *As the* rewardPending *member is cumulative, it is possible to exploit this behavior and artificially increase the pending rewards of a user. We advise that either a manual 0 assignment statement is introduced in the* emergencyWithdraw *function or a* delete *operation is conducted on the full struct located at* userInfo[msg.sender].

Alleviation: *The* emergencyWithdraw *function was properly fixed to zero out all members of the* UserInfo *struct.*

FINDING CATEGORIES

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

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

CHECKSUM CALCULATION METHOD

The “Checksum” field in the “Audit Scope” section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux “sha256sum” command against the target file.

WEBSITE
MHAUDITS.IO

TWITTER
@MHAUDITS