DeHacker

Code Security Assessment

PancakeSwap

April 6th, 2022

Contents

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Issue Categories

Every issue in this report was assigned a severity level from the following:

Critical severity issues

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

Major severity issues

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

Medium severity issues

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

Minor severity issues

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

Informational

A vulnerability that has informational character but is not affecting any of the code.

Overview

Project Summary

Project Name	PancakeSwap
Platform	Ethereum
website	https://pancakeswap.finance/
Language	Solidity
Codebase	https://github.com/pancakeswap/pancake-farm
Commit	05e7fbdd16a94b7e18b51811eda411fb4a1b4b41 daf8da084170ec8fe7ff705a7c0489dc8f730e50

Audit Summary

Delivery Date	April 6, 2022
Audit Methodology	Static Analysis, Manual Review

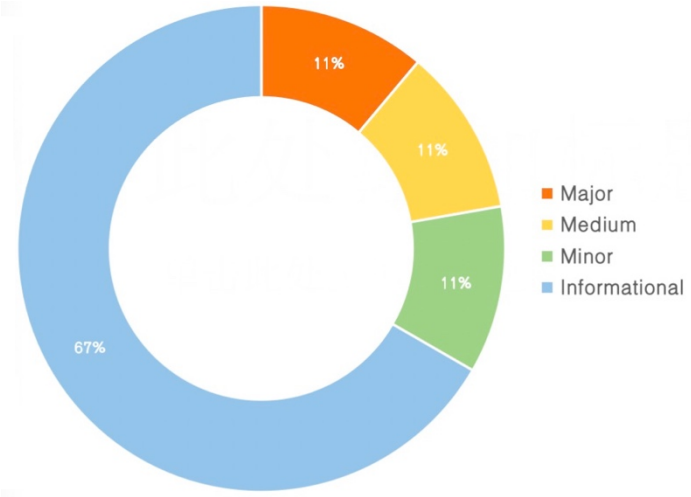
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	1	0	0	0	0	1
Medium	1	0	0	0	0	1
Minor	1	0	0	0	0	1
Informational	6	0	0	1	0	5
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
MCP	MasterChef.sol	61cda494749c052bc0abc18e9eee5d365924112c9b028a15b328d3615231183e
SCP	SousChef.sol	b0479b49ee6a067d6df573efc58b260b0b507ae033bf9ea77dccfad9316eed61
SBP	SyrupBar.sol	d3503231fb16984ce783991c27cd1d8270b6615677f7b002ba3a284911f5c48c

Findings



ID	Title	Category	Severity	Status
MCP-01	Variable Naming Convention	Coding Style	Informational	Acknowledged
MCP-02	Comment Typo	Coding Style	Informational	Resolved
MCP-03	Centralized Control of Bonus Multiplier	Logical Issue	Informational	Resolved
MCP-04	Assignment Optimization	Gas Optimization	Informational	Resolved
SBP-01	Incorrect Delegation Flow	Logical Issue	Major	Resolved
SBP-02	Inexistent Delegate Transfer	Logical Issue	Medium	Resolved
SCP-01	addressList Inaccuracy	Logical Issue	Minor	Resolved
SCP-02	Contract Purpose Unclear	Logical Issue	Informational	Resolved
SCP-03	Incorrect Reset Mechanism	Logical Issue	Informational	Resolved

Major

SBP-01 | Incorrect Delegation Flow

Category	Severity	Location	Status
Logical Issue	Major	SyrupBar.sol: 17	Resolved

Description

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

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.

Medium

SBP-02 | Inexistent Delegate Transfer

Category	Severity	Location	Status
Logical Issue	Medium	SyrupBar.sol: 1	Resolved

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.

Recommendation

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

Alleviation

After 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.

Minor

SCP-01 | addressList Inaccuracy

Category	Severity	Location	Status
Logical Issue	Minor	SousChef.sol: 120~122	Resolved

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.

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 events and blockchain analysis rather than contract storage.

Informational

MCP-01 | Variable Naming Convention

Category	Severity	Location	Status
Coding Style	Informational	MasterChef.sol: 71	Acknowledged

Description

The linked variables do not conform to the standard naming convention of Solidity whereby functions and variable names utilize the `camelCase` format unless variables are declared as `constant` in which case they utilize the `UPPER_CASE` format.

Recommendation

We advise that the naming conventions utilized by the linked statements are adjusted to reflect the correcttype of declaration according to the Solidity style guide

Alleviation

The PancakeSwap development team has acknowledged this exhibit but decided to not apply itsremediation in the current version of the codebase due to time constraints

MCP-02 | Comment Typo

Category	Severity	Location	Status
Coding Style	Informational	MasterChef.sol: 79	Resolved

Description

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

Recommendation

We advise that the comment text is corrected.

Alleviation

The comment typo was properly fixed.

MCP-03 | Centralized Control of Bonus Multiplier

Category	Severity	Location	Status
Logical Issue	Informational	MasterChef.sol: 113~115	Resolved

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.

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 withregards to all pool reward changes which are also first voted on by SYRUP holders through their votingportal using a Snapshot mechanism. This decentralizes the aspect of changing the multipliers via governance by SYRUP holders.

MCP-04 | Assignment Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	MasterChef.sol: 143	Resolved

Description

The linked statement will only yield a different output stored to totalAllocPoint only if the condition ofL146 yields true.

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.

SCP-02 | Contract Purpose Unclear

Category	Severity	Location	Status
Logical Issue	Informational	SousChef.sol: 1~156	Resolved

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.

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.

SCP-03 | Incorrect Reset Mechanism

Category	Severity	Location	Status
Logical Issue	Informational	SousChef.sol: 148~154	Resolved

Description

The `emergency Withdraw` 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.

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.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES ORMATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Coding Style

Coding Style findings usually do not affect the generated bytecode but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

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.

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.

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.

About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

BLOCKCHAIINS

Ethereum
Cosmos
Eos
Substrate

TECH STACK

Python
Solidity
Rust
c++

CONTACTS

https://dehacker.io
https://twitter.com/dehackerio
https://github.com/DeHacker-io/audits_public
https://t.me/dehackerio
https://blog.dehacker.io/

DeHacker

April 2022