CERTIK

Security Assessment

Pureswap

Apr 27th, 2021

Summary

This report has been prepared for Pureswap smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Dynamic Analysis, 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 produced by 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 recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations 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 given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Pureswap
Description	PureSwap is a decentralized exchange running on Binance Smart Chain with lots of other features that let you earn and win tokens.
Platform	BSC
Language	Solidity
Codebase	https://github.com/PureSwap2020/pureswap-contracts
Commits	1. b6d4e17aecf77ce435009088aba2ae42d925b1b9 2. a06b100eaa070a0e8e164b569adfd861800bda83

Audit Summary

Delivery Date	Apr 27, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Total Issues	11
● Critical	0
● Major	2
● Minor	0
● Informational	9
● Discussion	0

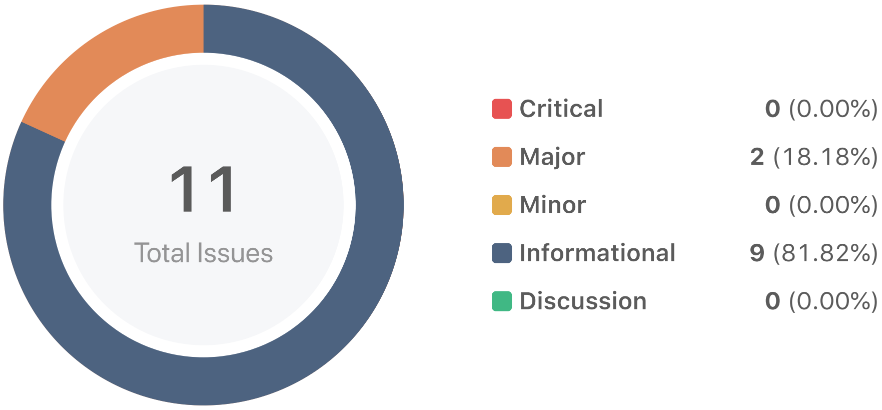
Audit Scope

ID	file	SHA256 Checksum
MPS	Migrator.sol	789abcc6943a8d369f64ef1ad68d699b60b602283b108d9a6db0922a2cf06bb4
MTP	MockToken.sol	e37a264653f8d5dcc7945a1b9980eabddf0c1467bfd34a3c09e03929ac2d2cb7
MUL	Multicall.sol	f834e013658162eb299c92e58ed96260fef150117bab398d46cc8a8411d60a88
OPS	Ownable.sol	c0c4a21fddd282c51e9e09a9c160c56811358b21738c12332700b9e6b2c1e759
PBP	PureBar.sol	8e8da40bc3e11c845a8ec0701aab09a57983555dad513bcaecb501a968703f0e
PCP	PureChef.sol	233e181e02c2c136de11a73b8ead57578bf5dc71d7da0e3e96352001c82c0e0e
PMP	PureMaker.sol	7797be47b15778154a414aa7d5072594a1ae6e4360271752172163f61e4c6d24
PRP	PureRoll.sol	145c706f8a04b0c03d9508a36159b88c4a648aa32146199e3520ee0ed365e0a1
PTP	PureToken.sol	0d2bf6ef00baa64a9c97c1f45a03361903d3dc49dc3dc3f5e79ab743a8f9f3be
SCP	SingleChef.sol	f62a0c34d88cf7a59d2c8ffe3dd7e7fae92543069ec83f65009d465bb9a74c76
TPS	governance/Timelock.sol	f6b905007d1a43539e30bcc046dce753cc5a129e5647eb7cb931c9d2e6228be0
IER	interfaces/IERC20.sol	89719e0ed1aa2069c48e6a1c5367d71c28e1ac6c1ac1fa3f055af1ad61ca5420
ISM	interfaces/ISwapMining.sol	df7f1a5c866bdf1d664b8011c072e8bdc47f5d00c604e86d74cb96c9ef81f15a
SER	libraries/SafeERC20.sol	a98b03567409156c916bfc7609964cfcf071101ad311d68382247cb1f48d8b08
SMP	libraries/SafeMath.sol	90233ebf8145f81dc990635fd72475720453cc1a8a2fcfbad2d4bc2f370f56b1
LIC	pureswap/LICENSE	2dc97585ae7643f0a97e14a0a55b83dc55ccadac9a331a9f916187887bc62bdf
PSE	pureswap/PureSwapERC20.sol	2508a1980b1701843e865df50bbbcb11c012be555b8ebc59a7a152503c6abff8
PSF	pureswap/PureSwapFactory.sol	af0a1c664f6d4180caff65c635af15eeb09373e138f428aa55bf38fa22b599b9

ID	file	SHA256 Checksum
PSP	pureswap/PureSwapPair.sol	87e140373945cd79d9b2de8b54fffab3ffec84dbf0b6bfcb218d8858555395e2
PSR	pureswap/PureSwapRouter.sol	604d923f7bc02b3a1a303f813c3504c88bd431e6dee1d154f57b1f11451c110f
REA	pureswap/README.md	5f2220d7ab2e62922b9b0f97f675f4c8e914a37a6dad039a942af203779c6d68
IEC	pureswap/interfaces/IERC20.sol	28fb844ef9c5a6e637ad879ecfdfb246746f02177305fba3e2c2fdd9a0931338
IUV	pureswap/interfaces/IUniswapV2Callee.sol	012a8b2ddcabe7f6b52fd13c6181ee124c15fdc9bcea797c1b8421252a9115d9
IUE	pureswap/interfaces/IUniswapV2ERC20.sol	40bdfa0a2cc99eaadb432b38547d6320e36d2bf3ac306c6b89ba676a5aa930c2
IUF	pureswap/interfaces/IUniswapV2Factory.sol	ffda2f4c85c5f1f6370a5877cdb95a8b9c78d13040b897d003ea6a061300eb7d
IUP	pureswap/interfaces/IUniswapV2Pair.sol	aa82bf61bb3dd028bf1f971a250f300a5b5ea704af067c94521d778d9abe2d76
IUR	pureswap/interfaces/IUniswapV2Router01.sol	40cb5c5bfc1d2be8b9ea0395871880f0c9f23e347a93c93f7c3a30a578ae7138
IUS	pureswap/interfaces/IUniswapV2Router02.sol	03b54a849ccc807ad66794d43f507bb0da5d2827dcbf7dee907a85afa2427bf3
IWE	pureswap/interfaces/IWETH.sol	1554ca91036a28ca29558556b0c51cff2d08845a7cd5349dcc4257955b45b68a
MAT	pureswap/libraries/Math.sol	c0e2137c4ae75e77a1b5c280467f910acd5c8cbf6d44f7ba98005c5d599e8ff3
SMS	pureswap/libraries/SafeMath.sol	962e662520db13530c001d73000111568f4f0f3db7b2c9ad8b80f75521722a1d

ID	file	SHA256 Checksum
THP	pureswap/libraries/TransferHelper.sol	24474660f6d91fbb161fc5b1ed7a2c7891cac209a6f3b45615518a2b75c60c0f
UQP	pureswap/libraries/UQ112x112.sol	6393d4879274cfceabb1b8c867ce8d61a25b47069f2732756c9d63bfbc54c647
UVL	pureswap/libraries/UniswapV2Library.sol	363ab3f48ad31281096224283fb1fe81bbb6df85395f3f2a064bda504e7d8f31

Findings



ID	Title	Category	Severity	Status
PCP-01	Lack of Input Validation	Volatile Code	● Informational	✓ Resolved
PCP-02	Unused Variable	Gas Optimization	● Informational	✓ Resolved
PCP-03	Administrator Capability	Optimization	● Major	ⓘ Acknowledged
PCP-04	Missing Emit Events	Volatile Code	● Informational	✓ Resolved
PMP-01	Zero Address to `bar`	Logical Issue	● Informational	✓ Resolved
PMP-02	Authority of Function `burn`	Control Flow	● Informational	✓ Resolved
PTP-01	Administrator Capability	Optimization	● Major	ⓘ Acknowledged
SCP-01	Lack of Input Validation	Volatile Code	● Informational	✓ Resolved
SCP-02	Unused Variable	Gas Optimization	● Informational	✓ Resolved
SCP-03	Economic Model of Reward	Logical Issue	● Informational	✓ Resolved
SCP-04	Missing Emit Events	Volatile Code	● Informational	✓ Resolved

PCP-01 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	● Informational	PureChef.sol: 87	✓ Resolved

Description

The assigned value to `_pureToken` should be verified as non zero value to prevent being mistakenly assigned as `address(0)` in the constructor of contract `PureChef.sol`. Violation of this may cause losing ownership of `_pureToken` authorization.

Recommendation

Check that the address is not zero by adding following checks in the constructor of contract `PureChef.sol`.

```
require(_pureToken != address(0), "_pureToken is zero address");
```

Alleviation

The development team heeded our advice and resolved this issue in commit a06b100eaa070a0e8e164b569adfd861800bda83.

PCP-02 | Unused Variable

Category	Severity	Location	Status
Gas Optimization	● Informational	PureChef.sol: 63	✓ Resolved

Description

The variable `BONUS_MULTIPLIER` is unused.

Recommendation

Consider removing the unused variable `BONUS_MULTIPLIER`.

Alleviation

The development team heeded our advice and resolved this issue in commit a06b100eaa070a0e8e164b569adfd861800bda83.

PCP-03 | Administrator Capability

Category	Severity	Location	Status
Optimization	● Major	PureChef.sol: 266~271	ⓘ Acknowledged

Description

To bridge the trust gap between the administrator and users, the administrator needs to express a sincere attitude with the consideration of the administrator team's anonymousness. The administrator has the responsibility to notify users with the following capability of the administrator:

- Administrator can transfer pure tokens to own account under unpredicted cases via `emergencyPureWithdraw` function

Examples:

```
266     function emergencyPureWithdraw() public onlyOwner {
267         uint balance = pureToken.balanceOf(address(this));
268         if (balance > 0) {
269             pureToken.transfer(msg.sender, balance);
270         }
271     }
```

Recommendation

Recommendation: The advantage of `emergencyPureWithdraw` function in the protocol is that the administrator reserves the ability to rescue the assets in this contract under unexpected cases. It is also worthy of note the downside of `emergencyPureWithdraw` function, where the treasury in this contract can be migrated to administrator's address.

To improve the trustworthiness of the project, any dynamic runtime changes on the protocol should be notified to clients. Any plan to call this `emergencyPureWithdraw` function is better to move to the execution queue of Timelock, and also emit event.

Alleviation

The development team responded as below: The owner will become a timelock contract after the system is stable.

PCP-04 | Missing Emit Events

Category	Severity	Location	Status
Volatile Code	● Informational	PureChef.sol: 120~132, 135~137, 140~149	✓ Resolved

Description

Several key actions are defined without event declarations. Example:

```
120 set()
```

```
135 setMigrator()
```

```
140 migrate()
```

Recommendation

Consider emitting events for key actions.

Alleviation

The development team heeded our advice and resolved this issue in commit a06b100eaa070a0e8e164b569adfd861800bda83.

PMP-01 | Zero Address to `bar`

Category	Severity	Location	Status
Logical Issue	● Informational	PureMaker.sol: 25, 46~54, 256	✓ Resolved

Description

The `bar` is assigned to be zero address in line 25 instead of being initialized in the constructor. It implies that the function `_toPURE` will transfer pureToken to zero address. This causes loss of pureToken.

Recommendation

Considering assigning valid value to `bar` in the constructor and adding self-explained comments if there are some special logic.

Alleviation

The development team responded as below: The pure received by PureMaker is to be destroyed. This is the design of the project. Part of the handling fee of dex repurchases pure and then destroys it.

PMP-02 | Authority of Function `burn`

Category	Severity	Location	Status
Control Flow	● Informational	PureMaker.sol: 259~262	✓ Resolved

Description

The sensitive function 'burn' can be called by anyone.

Recommendation

We advise developers to adopt `onlyOwner` modifier in openzeppelin to sensitive functions `burn` and add self-explained comments if there are some special logic.

Alleviation

The development team responded as below: Because the destruction of pure is a mechanical design, anyone can call the burn function to destroy it.

PTP-01 | Administrator Capability

Category	Severity	Location	Status
Optimization	● Major	PureToken.sol: 283~288	ⓘ Acknowledged

Description

To bridge the trust gap between the administrator and users, the administrator needs to express a sincere attitude with the consideration of the administrator team's anonymousness. The administrator has the responsibility to notify users with the following capability of the administrator:

- Administrator can transfer pure tokens to own account under unpredicted cases via emergencyWithdrawReward function

Examples:

```
283     function emergencyPureWithdraw() public onlyOwner {
284         uint256 bal = balanceOf(address(this));
285         if (bal > 0) {
286             transfer(msg.sender, bal);
287         }
288     }
```

Recommendation

Recommendation: The advantage of `emergencyPureWithdraw` function in the protocol is that the administrator reserves the ability to rescue the assets in this contract under unexpected cases. It is also worthy of note the downside of `emergencyPureWithdraw` function, where the treasury in this contract can be migrated to administrator's address.

To improve the trustworthiness of the project, any dynamic runtime changes on the protocol should be notified to clients. Any plan to call this `emergencyPureWithdraw` function is better to move to the execution queue of Timelock, and also emit event.

Alleviation

The development team responded as below: The owner will become a timelock contract after the system is stable.

SCP-01 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	● Informational	SingleChef.sol: 89	✓ Resolved

Description

The assigned value to `_pureToken` should be verified as non zero value to prevent being mistakenly assigned as `address(0)` in the constructor of contract `SingleChef.sol`. Violation of this may cause losing ownership of `_pureToken` authorization.

Recommendation

Check that the address is not zero by adding following checks in the constructor of contract `SingleChef.sol`.

```
require(_pureToken != address(0), "_pureToken is zero address");
```

Alleviation

The development team heeded our advice and resolved this issue in commit a06b100eaa070a0e8e164b569adfd861800bda83.

SCP-02 | Unused Variable

Category	Severity	Location	Status
Gas Optimization	● Informational	SingleChef.sol: 63	✓ Resolved

Description

The variable `BONUS_MULTIPLIER` is unused.

Recommendation

Consider removing the unused variable `BONUS_MULTIPLIER`.

Alleviation

The development team heeded our advice and resolved this issue in commit a06b100eaa070a0e8e164b569adfd861800bda83.

SCP-03 | Economic Model of Reward

Category	Severity	Location	Status
Logical Issue	● Informational	SingleChef.sol: 243~257, 221~240, 273~286	✓ Resolved

Description

This contract transfers reward pure tokens in the function `safePureTokenTransfer` that transfers the available reward of `dispathcer` to the user. It avoids the failure of `deposit` and `withdraw` when the reward pure tokens are insufficient, at the same time, the user may get less reward than expected after withdrawn pure tokens if there are no available pure tokens.

Alleviation

The development team responded as below: When the number of the dispatcher approve is taken away, it means the end of mining.

SCP-04 | Missing Emit Events

Category	Severity	Location	Status
Volatile Code	● Informational	SingleChef.sol: 98~100	✓ Resolved

Description

A key action is defined without event declaration. Example:

```
98  setDispatcher()
```

Recommendation

Consider emitting event for key action.

Alleviation

The development team heeded our advice and resolved this issue in commit a06b100eaa070a0e8e164b569adfd861800bda83.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that 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.

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.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

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

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in storage one.

Language Specific

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

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is 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 assessment. This report 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.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK 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.

CERTIK