CERTIK

Preliminary Comments

BonusCake

Oct 1st, 2021

Table of Contents

Summary

This report has been prepared for BonusCake to discover issues and vulnerabilities in the source code of the BonusCake 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 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;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	BonusCake
Platform	BSC
Language	Solidity
Codebase	https://bscscan.com/address/0xb84ddc645c27d4dc4bfa325c946f9d89d3afcc7a#code
Commit	

Audit Summary

Delivery Date	Oct 01, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

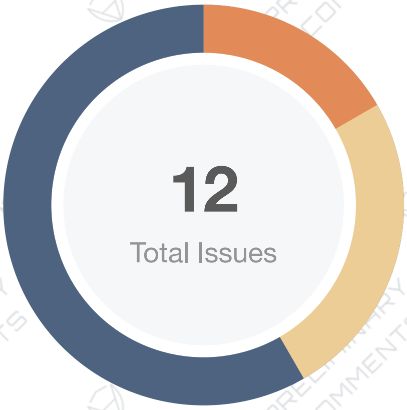
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0
● Major	2	2	0	0	0	0
● Medium	0	0	0	0	0	0
● Minor	3	3	0	0	0	0
● Informational	7	7	0	0	0	0
● Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum

Findings



🟥 **Critical**	**0** (0.00%)
🟧 **Major**	**2** (16.67%)
🟨 **Medium**	**0** (0.00%)
▨ **Minor**	**3** (25.00%)
🟦 **Informational**	**7** (58.33%)
🟩 **Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
BCC-01	Centralization Risk	**Centralization / Privilege**	● **Major**	ⓘ Pending
BCC-02	Potential Sandwich Attacks	Logical Issue	● Minor	ⓘ Pending
BCC-03	Third Party Dependencies	Volatile Code	● Minor	ⓘ Pending
BCC-04	Risk For Weak Randomness	Logical Issue	● Minor	ⓘ Pending
BCC-05	Missing Emit Events	Coding Style	● Informational	ⓘ Pending
BCC-06	Unreachable code	Coding Style	● Informational	ⓘ Pending
BCC-07	Irrelevant comments	Coding Style	● Informational	ⓘ Pending
BCC-08	Return value not handled	Volatile Code	● Informational	ⓘ Pending
BCC-09	Confusing naming	Coding Style	● Informational	ⓘ Pending
BCC-10	Unlocked Compiler Version	Language Specific	● Informational	ⓘ Pending
BCC-11	Minor typos	Coding Style	● Informational	ⓘ Pending
BCC-12	Centralized risk in `addLiquidity`	**Centralization / Privilege**	● **Major**	ⓘ Pending

BCC-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Major	bonuscake/BonusCake.sol	ⓘ Pending

Description

In the contract `BonusCake`, the role `owner` has the authority over the following functions:

- `updateDividendTracker()`
- `updateUniswapV2Router()`
- `excludeFromFees()`
- `BonusCakeexcludeFromDividends()`
- `excludeMultipleAccountsFromFees()`
- `setAutomatedMarketMakerPair()`
- `updateLiquidityWallet()`
- `updateGasForProcessing()`
- `updateClaimWait()`
- `updateL()`
- `updateS()`
- `updateM()`
- `updateLUCKHODL()`
- `updateSellFees()`
- `updateBuyFees()`
- `updateCAKEReward()`
- `updateProject()`
- `updateLiquidity()`
- `setAddress()`
- `setAddress()`

Regarding contract `BonusCakeDividendTracker`, the role `owner` has the authority over the following functions:

- `excludeFromDividends()`
- `updateClaimWait()`
- `getLuckAddress()`
- `setBalance()`

- `processAccount()`

Any compromise to the `owner` account may allow the hacker to take advantage of this and update the uniswap router (currently being the pancakeswap V2 router), exclude addressess from paying fees or receiving dividends amongst other critical actions.

Based on the record on chain, we could identify the owner as an EOA, and the owner address is `0x248cffa0524342eb47e6bffcd2abc169e5d65707` https://bscscan.com/address/0xb84ddc645c27d4dc4bfa325c946f9d89d3afcc7a#readContract.

Recommendation

We advise the client to carefully manage the `owner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

BCC-02 | Potential Sandwich Attacks

Category	Severity	Location	Status
Logical Issue	● Minor	bonuscake/BonusCake.sol: 422~440, 442~459	ⓘ Pending

Description

A sandwich attack might happen when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (before the transaction being attacked) a transaction to purchase one of the assets and make profits by backrunning (after the transaction being attacked) a transaction to sell the asset.

The following functions are called without setting restrictions on slippage or minimum output amount, so transactions triggering these functions are vulnerable to sandwich attacks, especially when the input amount is large:

- `swapTokensForEth()`
- `swapTokensForCake()`

Recommendation

We recommend setting reasonable minimum output amounts, instead of 0, based on token prices when calling the aforementioned functions.

BCC-03 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	● Minor	bonuscake/BonusCake.sol	ⓘ Pending

Description

The contract interacts with third party PancakeSwap protocols. The scope of the audit would treat those 3rd party entities as black boxes and assume its functional correctness. However in the real world, 3rd parties may be compromised that led to assets lost or stolen. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Recommendation

We understand that the business logic of BonusCake requires interaction with Pancakeswap. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

BCC-04 | Risk For Weak Randomness

Category	Severity	Location	Status
Logical Issue	● Minor	bonuscake/BonusCake.sol: 576~578	Pending

Description

The `_randomByModulus()` function is returning a pseudorandom number with `block.timestamp` and `block.difficulty`, and then generating the module of `numberOfTokenHolders`. The values of `block.timestamp`,`block.difficulty` and `numberOfTokenHolders` can be queried, so we think the returned value based on inner operations can be predicted. It is now well understood that the difficulty or timestamp is not a source of randomness; they can be manipulated if the attacker is also a miner.

```
function _randomByModulus(uint256 numberOfTokenHolders) private view returns(uint256) {
    return uint256(keccak256(abi.encodePacked(block.timestamp, block.difficulty))).mod(numberOfTokenHolders);
}
```

Recommendation

Consider obtaining random numbers based on a third-part random service such as Chainlink (https://docs.chain.link/docs/get-a-random-number/). According to Solidity Documentation, it is not recommended to use `block.timestamp`, `block.difficulty`, `gasLeft` or `blockhash` as a source of randomness (https://docs.soliditylang.org/en/latest/units-and-global-variables.html?highlight=block#block-and-transaction-properties).

BCC-05 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	● Informational	bonuscake/BonusCake.sol: 461~476, 442~459, 422~440, 309~311, 246~248, 242~244, 238~240, 234~236, 230~232, 226~228, 222~224, 218~220, 214~216, 210~212, 168~170, 180~183, 250~252, 254~256	Pending

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to customers.

- `BonusCakeexcludeFromDividends()`
- `setAutomatedMarketMakerPair()`
- `updateClaimWait()`
- `updateL()`
- `updateS()`
- `updateM()`
- `updateLUCKHODL()`
- `updateSellFees()`
- `updateBuyFees()`
- `updateCAKEReward()`
- `updateProject()`
- `updateLiquidity()`
- `setAddress()`
- `claim()`
- `swapTokensForEth()`
- `swapTokensForCake()`
- `addLiquidity()`

Recommendation

Consider adding events for sensitive actions, and emit them in the function.

BCC-06 | Unreachable code

Category	Severity	Location	Status
Coding Style	● Informational	bonuscake/BonusCake.sol: 537~539, 541~543	ⓘ Pending

Description

Given the require(false) statement, the code block from the functions `_transfer()` and `withdrawDividend()` of the `BonusCakeDividendTracker` contract will never be executed and is unnecessary.

Recommendation

We recommend removing the unreachable/unnecessary code block.

BCC-07 | Irrelevant comments

Category	Severity	Location	Status
Coding Style	● Informational	bonuscake/BonusCake.sol: 411	Pending

Description

The comment `this breaks the ETH -> HATE swap when swap+liquify is triggered` found in BonusCale.sol is not relevant to BonusCake.

```
        swapTokensForEth(half); // <- this breaks the ETH -> HATE swap when swap+liquify
is triggered
```

Recommendation

We recommend removing irrelevant comments.

BCC-08 | Return value not handled

Category	Severity	Location	Status
Volatile Code	● Informational	bonuscake/BonusCake.sol: 467~474	ⓘ Pending

Description

The return values of function `addLiquidityETH` are not properly handled.

```
        // add the liquidity
        uniswapV2Router.addLiquidityETH{value: ethAmount}(
            address(this),
            tokenAmount,
            0, // slippage is unavoidable
            0, // slippage is unavoidable
            liquidityWallet,
            block.timestamp
        );
```

Recommendation

We recommend using variables to receive the return value of the functions mentioned above and handle both success and failure cases if needed by the business logic.

BCC-09 | Confusing naming

Category	Severity	Location	Status
Coding Style	● Informational	bonuscake/BonusCake.sol: 250~256	ⓘ Pending

Description

The BonusCake smart contract had two different `setAddress` functions with different arguments that would do a completely different task from each other. The first one (with just one argument) would update the `projectAddress` (currently the owner of the project `0x248cFFa0524342eB47E6bfFcD2abc169E5D65707`):

```
function setAddress(address payable projectAddress) public onlyOwner {
  _projectAddress = projectAddress;
}
```

While the second one, with 2 arguments, would add/remove the `addr` address to/from the blacklist depending on the value of the boolean `isBlack`:

```
function setAddress(address addr, bool isBlack) public onlyOwner {
  blacklist[addr] = isBlack;
}
```

Given the generic naming used and the different purpose of both functions, it is believed that the function names could lead to confusion.

Recommendation

We would recommend to adjust the naming or the logic to the function intentions.

BCC-10 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	● Informational	bonuscake/BonusCake.sol: 16	ⓘ Pending

Description

The `BonusCake` and `BonusCakeDividendTracker` contracts have unlocked compiler version. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

The following pragma statement is used:

```
pragma solidity ^0.6.2;
```

Recommendation

We advise that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.6.2` the contract should contain the following line:

```
pragma solidity 0.6.2;
```

BCC-11 | Minor typos

Category	Severity	Location	Status
Coding Style	● Informational	bonuscake/BonusCake.sol: 58	ⓘ Pending

Description

The spelling of 'exlcude' is a typo

Recommendation

Change it to 'exclude'

BCC-12 | Centralized risk in `addLiquidity`

Category	Severity	Location	Status
Centralization / Privilege	● Major	bonuscake/BonusCake.sol: 467~473	ⓘ Pending

Description

```
        uniswapV2Router.addLiquidityETH{value: ethAmount}(
            address(this),
            tokenAmount,
            0, // slippage is unavoidable
            0, // slippage is unavoidable
            liquidityWallet,
            block.timestamp
        );

```

The `addLiquidity` function calls the `uniswapV2Router.addLiquidityETH` function with the `to` address specified as `liquidityWallet` for acquiring the generated LP tokens from the `BonusCake-BNB` pool. As a result, over time the `liquidityWallet` address will accumulate a significant portion of LP tokens.If the `liquidityWallet` is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Recommendation

We advise the `to` address of the `uniswapV2Router.addLiquidityETH` function call to be replaced by the contract itself, i.e. `address(this)`, and to restrict the management of the LP tokens within the scope of the contract’s business logic. This will also protect the LP tokens from being stolen if the `liquidityWallet` account is compromised. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;
- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

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.

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.

Volatile Code

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

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 but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

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.

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 you ("Customer" or 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, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

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.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

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