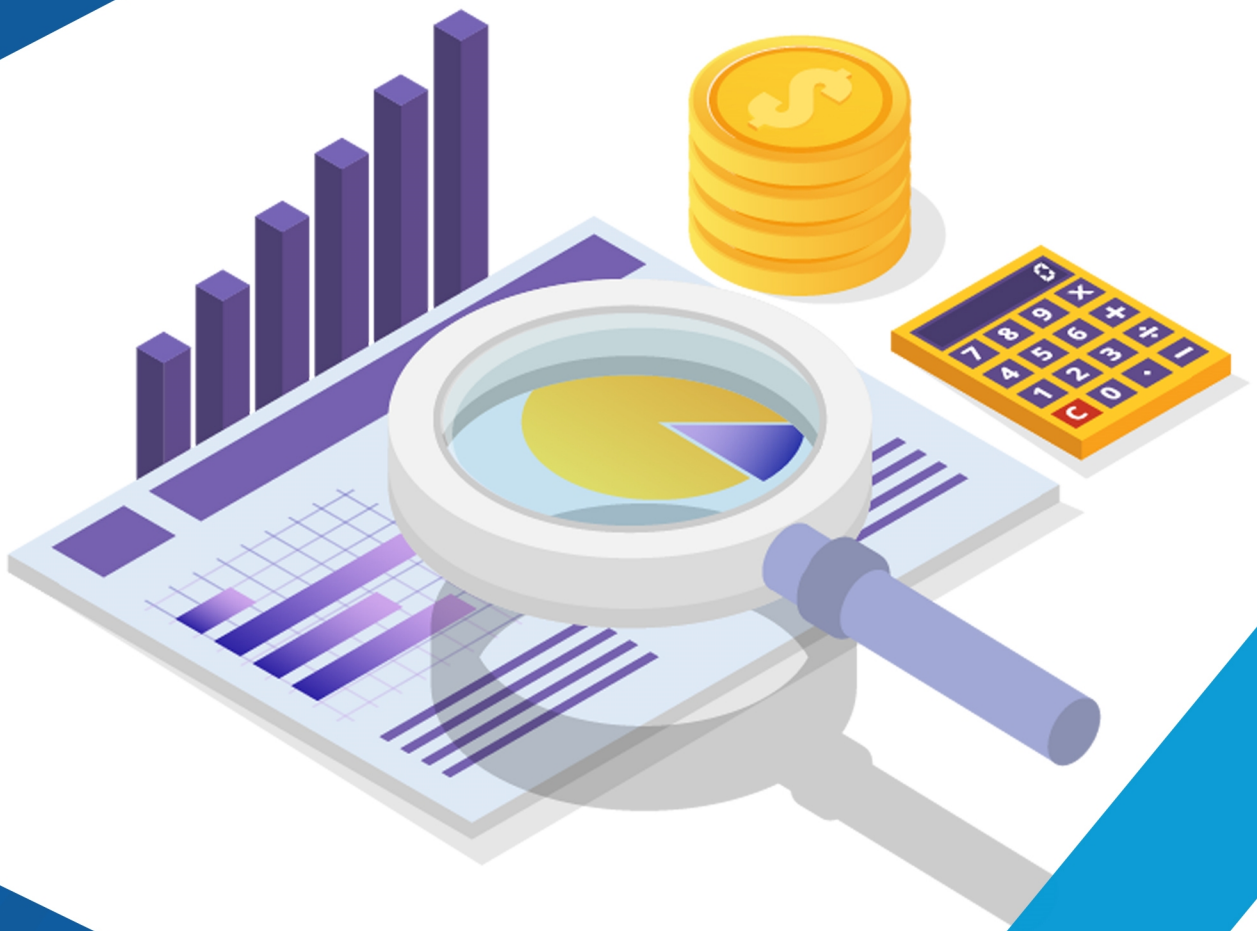
QuillAudits



Audit Report
December, 2020

CHOCOSWAP

Contents

Introduction

This Audit Report mainly focuses on the overall security of some of the Chocoswap Smart Contracts. With this report, we have tried to ensure the reliability and correctness of their smart contract by rigorous assessment of their system's architecture and the smart contract codebase.

Auditing Approach and Methodologies applied

The Quillhash team has performed rigorous testing of the project starting with analysing the code design patterns in which we reviewed the smart contract architecture to ensure it is structured and safe use of third party smart contracts and libraries.

Our team then performed a formal line by line inspection of the Smart Contract to find any potential issue like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.

The code was tested in collaboration of our multiple team members and this included -

- Testing the functionality of the Smart Contract to determine proper logic has been followed throughout the whole process.
- Analysing the complexity of the code in depth and detailed, manual review of the code, line-by-line.
- Deploying the code on testnet using multiple clients to run live tests.
- Analysing failure preparations to check how the Smart Contract performs in case of any bugs and vulnerabilities.
- Checking whether all the libraries used in the code are on the latest version.
- Analysing the security of the on-chain data.

Audit Details

This audit is based of commit hash
`70240143695f37cd11a65bbd70a8ea4949d67472` of the
GitHub repository - https://github.com/chocoswap/chocoswap/

The following contracts are in scope for this audit:
1. Choco.sol
2. ChocoPresale.sol
3. VNLA.sol
4. VNLADistribution.sol

External dependencies and any other contract is not in scope for this audit.

Audit Goals

The focus of the audit was to verify that the Smart Contract System is secure, resilient and working according to the specifications. The audit activities can be grouped in the following three categories:

Security
Identifying security related issues within each contract and the system of contract.

Sound Architecture
Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.

Code Correctness and Quality
A full review of the contract source code. The primary areas of focus include:
- Accuracy
- Readability
- Sections of code with high complexity

Issue Categories

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

High severity issues

Issues on this level are critical to the smart contract's performance/ functionality and should be fixed before moving to a live environment.

Medium severity issues

Issues on this level could potentially bring problems and should eventually be fixed.

Low severity issues

Issues on this level are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

Number of issues per severity

	Low	Medium	High
Open	7	1	0
Closed	1	2	0

Manual Audit

Low severity issues

1. Source code for the `Migrator` contract is not provided. It is a critical dependency of `VNLADistribution` contract but it is not part of the audit and the source code was not provided to us. It is recommended to get the migrator contract audited separately. A bug in the `Migrator` contract can lead to exploits in the `VNLADistribution` contract.

2. In `chocoPresale` contract, the system does not check if the investment amount is bigger than phase 2 limit when the contract is in phase 1. Therefore, it is possible to invest more than the limit if the investment is made while the sale is still in the phase 1.

3. The contracts lack documentation and code comments. Verifying business logic without documented requirements is not possible.

4. In the `chocoPresale` contract, the `onlyOwner` modifier can be used on the `transfer` and the `mint` function instead of manually checking that the `msg.sender` is the owner.

5. The fallback function can be used instead of `receive` in the `chocoPresale` contract.

6. The codebase is missing the `package.json`. It is hard to verify the dependency versions that are intended to be used without this file.

   Status: Fixed

7. `preSalePhase` can be made an enum instead of `uint256` for clarity.

8. `pChoc` token of `chocoPresale` contract is not ERC20 compliant since it is missing functions like `approve`. Also, only the owner can transfer the `pChoc` token.

Medium severity issues

1. The owner controls multiple critical aspects of the system. For example, the owner can mint an unlimited number of `VNLAToken`, add malicious migrator and lp tokens in `VNLADistribution`, and also mint `pChoc` after the presale is over. It is recommended to create a DAO and make it the owner or limit the powers of the owner.

   Status: Acknowledged

2. It is possible to add duplicate pools in the `VNLADistribution` contract. The contract is not made to handle duplicate pools and this can result in critical malfunctions. However, only the owner can add pools and hence this is not a critical issue. In any case, it is recommended to check for duplicates in the `_add` function to avoid any accidents.

   Status: Fixed

3. Reentry attacks are possible in `withdraw`, `deposit`, `emergencyWithdraw` and, `updatePool` functions of the `VNLADistribution`. These attacks can result in drainage of funds from the contract. However, these can happen only when either a lp pool or the migrator is malicious. Since only the owner can add pools and migrator, this is not a critical issue. However, it is recommended to modify the contracts to do the storage changes first and then call external contracts. This pattern is commonly known as the checks-effects-interactions pattern.

   Status: Fixed

High severity issues

No high severity issues

Automated Audit

SmartCheck

Smartcheck is a tool for automated static analysis of Solidity source code for security vulnerabilities and best practices. SmartCheck translates Solidity source code into an XML-based intermediate representation and checks it against XPath patterns. Smartcheck shows significant improvements over existing alternatives in terms of false discovery rate (FDR) and false negative rate (FNR). It gave the following result for the relevant contracts:
https://tool.smartdec.net/scan/c116dd877c564d3692a6afb7a2467c74

Slither

Slither, an open-source static analysis framework. This tool provides rich information about Ethereum smart contracts and has the critical properties. While Slither is built as a security-oriented static analysis framework, it is also used to enhance the user's understanding of smart contracts, assist in code reviews, and detect missing optimizations.

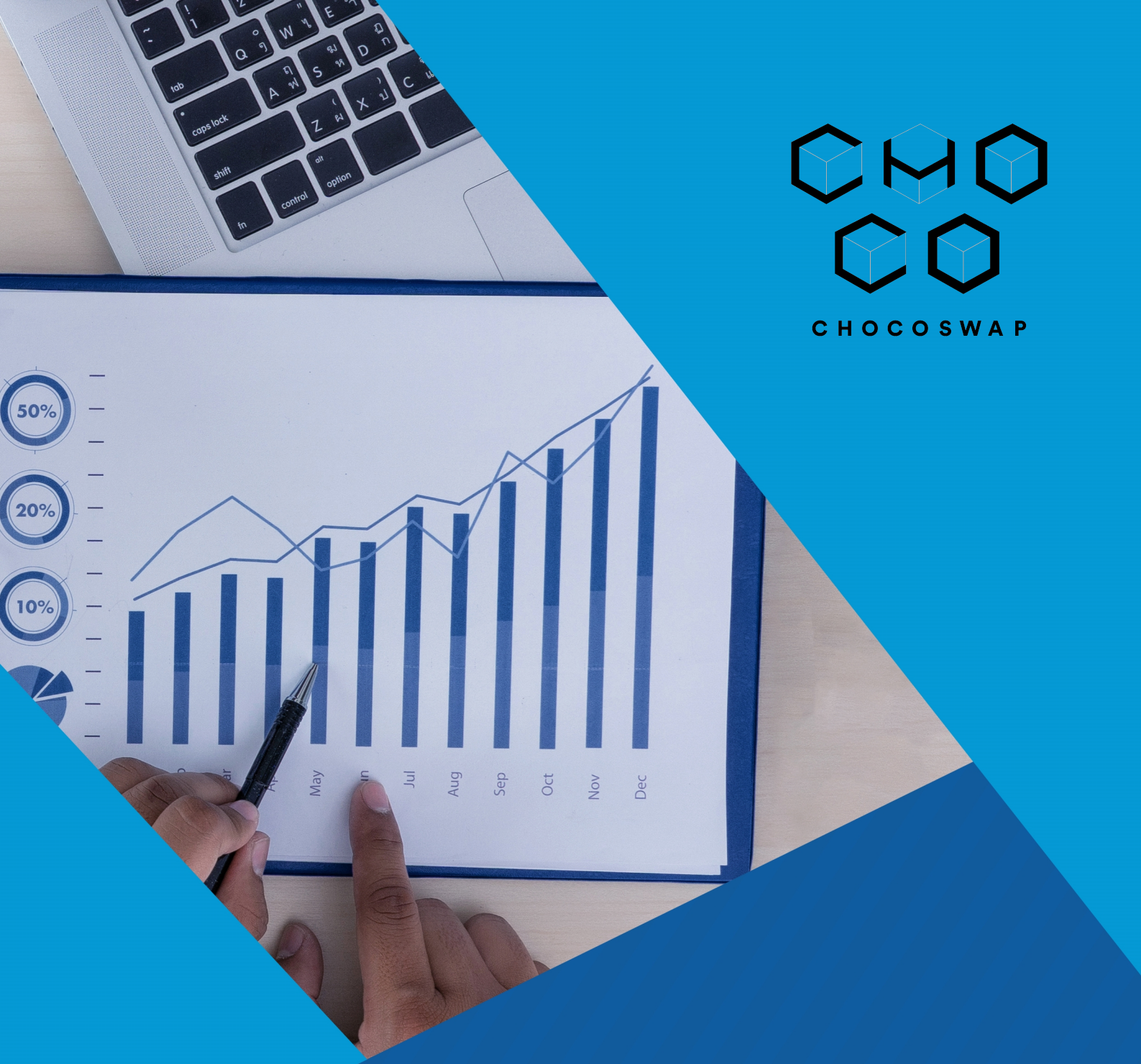
The output generated by Slither can be found at
https://gist.github.com/maxsam4/d62b51f9f099b24bd5434bf572fed5e5

Disclaimer

This report is not an endorsement or indictment of any particular project or team, and the report does not guarantee the security of any particular project. This report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. This report does not provide any warranty or representation to any Third-Party in any respect, including regarding the bugfree nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on this report in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. I owe no duty to any Third-Party by virtue of publishing these Reports.

The scope of my review is limited to a review of Solidity code and only the Solidity code noted as being within the scope of the review within this report. The Solidity language itself 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 Solidity that could present security risks. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty.

This audit does not give any warranties on finding all possible security issues of the given smart contracts, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues. As one audit-based assessment cannot be considered comprehensive, I always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.



CHOCOSWAP

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

hello@quillhash.com