SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Deficliq
Date: November 12th, 2020
Platform: Ethereum
Language: Solidity

This document may contain confidential information about IT systems and the intellectual property of the customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the customer, or it can be disclosed publicly after all vulnerabilities fixed - upon a decision of the customer.

Document

Name	Smart Contract Code Review and Security Analysis Report for Deficliq(11 pages).
Type	ERC-20 token
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Approved by	Andrew Matiukhin | CTO and co-founder Hacken
Archive-name	cliq.zip
SHA-1 hash	97ff586cce85aa9e339ecf9e25760e13424985b3
Contract Address	https://etherscan.io/address/0x0Def8d8addE14c9eF7c2a986dF3eA4Bd65826767#code
Contract Creator(Owner)	0xbb3f8f2774729b17e2abc8be6bc6383acac0d5da
Contract Minner	0x2a216a0dbb6489086401e77f3d04461b96fb19eb
Timeline	9th NOV 20 -12th NOV 2020
Changelog	12th NOV 2020 - Initial Audit 16th NOV 20202 - Contract address added, code validated

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Deficliq (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer`s smart contract and its code review conducted between November 9th, 2020 - November 12th, 2020.

Scope

The scope of the project is smart contracts in the repository:

1\ Archive - cliq.zip
File SHA-1 hash - 97ff586cce85aa9e339ecf9e25760e13424985b3

Contract	In scope
cliq.sol ERC20.sol SafeMath.sol MinterRole.sol Roles.sol ERC20Detailed.sol ERC20Capped.sol Context.sol ERC20Burnable.sol IERC20.sol ERC20Mintable.sol	Yes

2\ Deploy address:
https://etherscan.io/address/0x0Def8d8addE14c9eF7c2a986dF3eA4Bd65826767

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference

	Implicit visibility level Deployment Consistency Repository Consistency Data Consistency
Functional review	Business Logics Review Functionality Checks Access Control & Authorization Escrow manipulation Token Supply manipulation Assets integrity User Balances manipulation Data Consistency manipulation Kill-Switch Mechanism Operation Trails & Event Generation

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations, and info statements can't affect smart contract execution and can be ignored.

Executive Summary

According to the assessment, Customer`s smart contracts are well-secured.

Insecure
Poor secured
Secured
Well-secured
You are here

Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in the AS-IS section, and all found issues can be found in the Audit overview section.

We found **1** low issue in smart contract code.

AS-IS overview

CLIQ

Description

CLIQ is an ERC-20 token implementation.

Imports

CLIQ contract has following imports:

- ERC20Detailed.sol
- ERC20Burnable.sol
- ERC20Capped.sol

All of them are imported from OpenZeppelin library

@openzeppelin/contracts/token/ERC20/

Inheritance

CLIQ is ERC20Detailed, ERC20Burnable, ERC20Capped*.*

Usages

CLIQ contract has no custom usages.

Structs

CLIQ contract has no custom data structures.

Enums

CLIQ contract has no custom enums.

Events

CLIQ contract has no custom events.

Modifiers

CLIQ contract has no custom modifiers.

Fields

CLIQ contract has no custom fields.

Functions

CLIQ has following functions:

- ***constructor***

Description

CLIQ constructor calls ERC20Capped, ERC20Detailed constructors and mints initial balance to deplyer

Visibility

public

Input parameters

- string memory name
- string memory symbol
- uint8 decimals
- uint256 cap
- uint256 initialSupply

Constraints

None

Events emit

None

Output

None

Audit overview

🟥 🟥 🟥 🟥 Critical

No critical issues were found.

🟨 🟨 🟨 High

No high issues were found.

🟩 🟩 Medium

No medium issues were found.

🟦 Low

1. Pragma version is not locked: pragma solidity ^0.5.0;

   It’s highly recommended to lock pragma to the latest stable solidity version.

🟦 Lowest / Code style / Best Practice

No best practice issues were found.

Conclusion

Smart contract within the scope was manually reviewed and analyzed with static analysis tools. For the contract high level description of functionality was presented in AS-IS overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security engineers found **1** low issue during the audit. The code is clean; follows smart contract security best practice.

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status, or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on blockchain platform. The platform, its programming language, and other software related to the smart contract can have their vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee the explicit security of the audited smart contracts.