CERTIK

CertiK Audit Report for GXC

CertiK Audit Report for

GXC

Request Date: 2020-08-24

Revision Date: 2020-09-04

Platform Name: GXChain

Contents

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 Verification Services Agreement between CertiK and GXC (the "Company"), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, 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.

About CertiK

CertiK is a technology-led blockchain security company founded by Computer Science professors from Yale University and Columbia University built to prove the security and correctness of smart contracts and blockchain protocols.

CertiK, in partnership with grants from IBM and the Ethereum Foundation, CertiK's mission of every audit is to apply different approaches and detection methods, ranging from manual, static, and dynamic analysis, to ensure that projects are checked against known attacks and potential vulnerabilities. CertiK leverages a team of seasoned engineers and security auditors to apply testing methodologies and assessments to each project, in turn creating a more secure and robust software system.

CertiK has served more than 100 clients with high quality auditing and consulting services, ranging from stablecoins such as Binance's BGBP and Paxos Gold to decentralized oracles such as Band Protocol and Tellor. CertiK customizes its engineering tool kits, while applying cutting-edge research on smart contracts, for each client on its project to offer a high quality deliverable. For more information: https://certik.io.

Executive Summary

This report has been prepared for **GXC** to discover issues and vulnerabilities in the source code of their **ERC 20 Token** 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.

Testing Summary

SECURITY LEVEL

Smart Contract Audit

This report has been prepared as a product of the Smart Contract Audit request by GXC.

This audit was conducted to discover issues and vulnerabilities in the source code of GXC's ERC 20 token.

TYPE	Smart Contract
SOURCE CODE	https://github.com/gxchain/gxc-relay-contract
PLATFORM	GXChain
LANGUAGE	C++/Solidity
REQUEST DATE	Aug 24, 2020
DELIVERY DATE	Sep 04, 2020
METHODS	A comprehensive examination has been performed using Dynamic Analysis, Static Analysis, and Manual Review.

Review Notes

Introduction

CertiK team was contracted by the GXC team to audit the design and implementation of their ERC 20 token smart contract.

The audited source code link is:

- relay Source Code: https://github.com/gxchain/gxc-relay-contract/commit/a867ad433a48ca0794ca50d3f9fc36baba5e8b95
- Source Code SHA-256 Checksum

  relay.cpp (commit a867ad433a48ca0794ca50d3f9fc36baba5e8b95)

  hash 291b900fd0372872d6a92bfe29739477d8025422130db9386928bdc4cab93cb6
- GXC Source Code: https://github.com/gxchain/gxc-relay-contract/commit/5bcd84ec56e05c49e78826d0fdd97a18a2115bf6
- Source Code SHA-256 Checksum

  GXC.sol (commit 5bcd84ec56e05c49e78826d0fdd97a18a2115bf6)

  hash 873db0972c23682140f08f4015564404a1741673a30967d5a12be4b311817691

The goal of this audit was to review the Solidity implementation for its business model, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

The findings of the initial audit have been conveyed to the team behind the contract implementations and the source code is expected to be re-evaluated before another round of auditing has been carried out.

Documentation

The sources of truth regarding the operation of the contracts in scope were lackluster and **are something we advise to be enriched to aid in the legibility of the codebase as well as project**. To help aid our understanding of each contract's functionality we referred to in-line comments and naming conventions.

These were considered the specification, and when discrepancies arose with the actual code behaviour, we consulted with the GXC team or reported an issue.

Summary

The codebase of the project is a typical ERC implementation and the locking mechanism of the token is derived from an officially recognized library, specifically from OpenZeppelin.

Certain optimization steps that we pinpointed in the source code mostly referred to coding standards and inefficiencies, however **1 minor vulnerability was identified during our audit that solely concerns the specification**. The codebase of the project strictly adheres to the standards and interfaces imposed by the OpenZeppelin open-source libraries and **can be deemed to be of high security and quality**.

Certain discrepancies between the expected specification and the implementation of it were identified and were relayed to the team, however they pose no type of vulnerability and concern an optional code path that was unaccounted for.

Recommendations

Overall, the codebase of the contracts should be refactored to assimilate the findings of this report, enforce linters and / or coding styles as well as correct any spelling errors and mistakes that appear throughout the code **to achieve a high standard of code quality and security**.

Findings

relay.cpp (**commit** a867ad433a48ca0794ca50d3f9fc36baba5e8b95)

Exhibit 1

TITLE	TYPE	SEVERITY	LOCATION
Unchecked function parameter	Optimization	Informational	relay.cpp: L39

[INFORMATIONAL] Description:

The parameter “addr” is not verified before usage.

Recommendations:

We recommend to add:

graphene_assert(!addr.empty(), "Address is not valid");

(GXC - resolved) The issue is addressed in commit

66a0b2dc97e52c4b6e64c93f9ad04dd793a1ce1d.

Exhibit 2

TITLE	TYPE	SEVERITY	LOCATION
Unchecked function parameter	Optimization	Informational	relay.cpp: L55

[INFORMATIONAL] Description:

The parameter “from_account” is not verified before usage.

Recommendations:

(GXC - confirmed) The parameter “from_account” is an address from ethereum, not able to be fetched and checked.

So we recommend to change to below codes:

graphene_assert(from_account.size() > 0, "Invalid account_name from_account");

Exhibit 3

TITLE	TYPE	SEVERITY	LOCATION
Absence of necessary error message	Optimization	Informational	relay.cpp: L56

[INFORMATIONAL] Description:

If the ”from_target“ is not "ETH", there is no error message and the function will end.

Recommendations:

We recommend to add:

graphene_assert (from_target == "ETH","Invalid chain name, only support ETH so far");

(GXC - resolved) The issue is addressed in commit

66a0b2dc97e52c4b6e64c93f9ad04dd793a1ce1d.

Exhibit 4

TITLE	TYPE	SEVERITY	LOCATION
Absence of necessary error message	Optimization	Informational	relay.cpp: L100

[INFORMATIONAL] Description:

If the “target” is not "ETH", there is no error message and the function will end.

Recommendations:

We recommend to add:

graphene_assert (target == "ETH","Invalid chain name, only support ETH so far");

(GXC - resolved) The issue is addressed in commit

66a0b2dc97e52c4b6e64c93f9ad04dd793a1ce1d.

Exhibit 5

TITLE	TYPE	SEVERITY	LOCATION
Misleading error message	Optimization	Informational	relay.cpp: L130

[INFORMATIONAL] Description:

The error message is not accurate.

Recommendations:

We recommend to use :

graphene_assert(idx != fund_out_table.end(), "There is nothing to withdraw");

(GXC - resolved) The issue is addressed in commit

66a0b2dc97e52c4b6e64c93f9ad04dd793a1ce1d.

GXC.sol (**commit** 5bcd84ec56e05c49e78826d0fdd97a18a2115bf6)

Exhibit 6

TITLE	TYPE	SEVERITY	LOCATION
Compiler version	Optimization	Minor	GXC.sol: L3

[Minor] Description:

The compiler version utilized throughout the project uses the "^" prefix specifier, denoting that a compiler at or above the version included after the specifier should be used to compile the contracts. Also, the compiler version should be consistent throughout the codebase.

Recommendations:

We recommend choosing a certain version.

(GXC - resolved) The issue is addressed in commit 79d90eb58f45e6bb4186113e776b3a6d50831792.

Exhibit 7

TITLE	TYPE	SEVERITY	LOCATION
Gas consumption	Optimization	Informational	GXC.sol: L13 & L19

[INFORMATIONAL] Description:

To avoid gas consumption, unchanged variables 'arrayLength' & 'decimals_' can be defined as constant.

Recommendations:

We recommend defining unchanged variables “arrayLength” & “decimals_” as constants.

(GXC - resolved) The issue is addressed in commit 66a0b2dc97e52c4b6e64c93f9ad04dd793a1ce1d.

Exhibit 8

TITLE	TYPE	SEVERITY	LOCATION
Gas consumption	Optimization	Informational	GXC.sol: L76

[INFORMATIONAL] Description:

Consider adding a function state modifier in the function definition of function “getParams()”.

Recommendations:

We recommend changing it as follows:

function getParams() public view returns (uint256 ,uint256){...}

(GXC - resolved) The issue is addressed in commit

f7a3ad0e50018daf03e3d94db99a0c42c7fd60f1.

Exhibit 9

TITLE	TYPE	SEVERITY	LOCATION
Gas consumption	Optimization	Informational	GXC.sol: L80

[INFORMATIONAL] Description:

Consider adding a function state modifier in the function definition of function “getTxids()”.

Recommendations:

We recommend changing it as follows:

function getTxids() public view returns (string[10] memory) {...}

(GXC - resolved) The issue is addressed in commit

f7a3ad0e50018daf03e3d94db99a0c42c7fd60f1.