CERTIK

CertiK Audit Report for Golff Finance

CertiK Audit Report for

GOLFF FINANCE

Request Date: 2020-09-03

Revision Date: 2020-09-06

Platform Name: EVM

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 Golff Finance (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 **GOLFF** 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 Golff finance.
This audit was conducted to discover issues and vulnerabilities in the source code of Golff's ERC 20 token.

TYPE	Smart Contract
SOURCE CODE	https://github.com/golff-protocol/golff-pool
PLATFORM	EVM
LANGUAGE	Solidity
REQUEST DATE	Sep 03, 2020
DELIVERY DATE	Sep 06, 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 **Golff** team to audit the design and implementation of their ERC 20 token smart contract.

The audited source code link is:

- pool Source Code:

  https://github.com/golff-protocol/golff-pool/blob/master/contracts/pool/

  commit 68d379beb43762e49de69f739dddcc3c8564fde9

  Source Code SHA-256 Checksum

  GOFETHPOOL.sol hash

  3ce942635dfff7d838e33f368c4e044137c0fa7e2cbc2eb1cbf06a5fa179d7c1

  GOFGolffPool.sol hash

  cfa9da273aa7b612fa4bf8f980ade24d85eec6985946f5f9be52c8ad58414b38

  GOFLINKPool.sol hash

  1a84aee05120b5625499312f4589a30a472bb3f599e2fbe5f0e84072b5ecf417

  GOFYFIIPool.sol hash

  dccaaaddfe2c940a067d740dc3f8ae224aa641d490ae66f564ef02e48c1846b3

  GOFHTPool.sol hash

  3557d5f8d3893910f4af37d93e554ee85bdfe81b97a04fc20ddc09b7d623148e

  GOFUSDTPool.sol hash

  a1cdfd7aed803d4b60b7ceda41de4a31b958cbc89cfbeac629a683555508bcd9

- token Source Code:

  https://github.com/golff-protocol/golff-pool/blob/master/contracts/token/GOF.sol

  commit 68d379beb43762e49de69f739dddcc3c8564fde9

  Source Code SHA-256 Checksum

  GOF.sol hash

  d49f63617ab901f8d7466b878bdd9534dffb8710c5ed2ab0a8d02b42c0f88c91

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 **Golff** 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 major 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

Exhibit 1

TITLE	TYPE	SEVERITY	LOCATION
Unlocked Compiler Version Declaration	Optimization	Informational	GOFETHPOOL.sol, GOFYFIIPool.sol, GOFUSDTPool.sol, GOFLINKPool.sol, GOFHTPool.sol, GOFGolffPool.sol, GOF.sol

[INFORMATIONAL] Description:

The compiler version utilized throughout the project uses the “^” prefix specifier, denoting that a compiler version which is greater than the version will be used to compile the contracts. Recommend the compiler version should be consistent throughout the codebase.

Recommendations:

It is a general practice to instead lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and be able to identify ones more easily. We recommend locking the compiler at the lowest possible version that supports all the capabilities wished by the codebase. This will ensure that the project utilizes a compiler version that has been in use for the longest time and as such is less likely to contain yet-undiscovered bugs.

(Golff - resolved) The issue is addressed in commit

1b14d1872e56527c033ae7750999b9c311ffe65b.

Exhibit 2

TITLE	TYPE	SEVERITY	LOCATION
Gas consumption	Optimization	Informational	GOFETHPOOL.sol, GOFYFIIPool.sol, GOFUSDTPool.sol, GOFLINKPool.sol, GOFHTPool.sol, GOFGolffPool.sol: L596

[INFORMATIONAL] Description:

Variable “startTime” is never changed, better to define it as constant to avoid gas consumption.

Recommendations:

Consider changing variable “startTime” as a constant.

```
uint256 public constant startTime = 1598534100;
```

(Golff - resolved) The issue is addressed in commit

986231c9d3593fc5f489e66c18a7ff228b4907a3.

Exhibit 3

TITLE	TYPE	SEVERITY	LOCATION
Variable not initialised	Optimization	Informational	GOFETHPOOL.sol, GOFYFIIPool.sol, GOFUSDTPool.sol, GOFLINKPool.sol, GOFHTPool.sol, GOFGolffPool.sol: L600

[INFORMATIONAL] Description:

The variable “rewardPerTokenStored” is not initialized before usage.

It can be used in the function “rewardPerToken” before initialization.

Recommendations:

Consider changing it as following:

`uint256 public rewardPerTokenStored = 0;`

(Golff - resolved) The issue is addressed in commit

986231c9d3593fc5f489e66c18a7ff228b4907a3.

Exhibit 4

TITLE	TYPE	SEVERITY	LOCATION
Usage of modifier is abused	Optimization	Informational	GOFETHPOOL.sol, GOFYFIIPool.sol, GOFUSDTPool.sol, GOFLINKPool.sol, GOFHTPool.sol, GOFGolffPool.sol:L611

[INFORMATIONAL] Description:

Modifiers can be used to change the behavior of functions.

The “updateReward()” modifier looks like a function rather than a modifier.

Recommendations:

Consider changing it as a function and using it wherever it is needed.

Exhibit 5

TITLE	TYPE	SEVERITY	LOCATION
Missing definition of constant variable	Optimization	Informational	GOFETHPOOL.sol, GOFYFIIPool.sol, GOFUSDTPool.sol, GOFLINKPool.sol, GOFHTPool.sol, GOFGolffPool.sol: L637,L646,L729

[INFORMATIONAL] Description:

As known, Wei is the smallest denomination of ether, an ether is 10^{18} Wei, but there are still other decimals of tokens on the ethereum. It could be easier to use if we define it as a constant variable.

Recommendations:

We recommend defining “1e18 ” as a constant variable and replacing it at L637 , L646 and L730.

(Golff - resolved) The issue is addressed in commit 0a1582141c26b0a4f8f486d441c51aedd36fd729.

Exhibit 6

TITLE	TYPE	SEVERITY	LOCATION
Gas consumption	Optimization	Informational	GOFETHPOOL.sol, GOFYFIIPool.sol, GOFUSDTPool.sol, GOFLINKPool.sol, GOFHTPool.sol, GOFGolffPool.sol: L650,L656,L667

[INFORMATIONAL] Description:

Incorrect order of modifiers could cause an accident, but in our case, it may take more gas than we want if these functions are called before the pool starts.

Recommendations:

We recommend moving the “checkStart” ahead of “updateReward”.

(Golff - resolved) The issue is addressed in commit 815854347fdf2e04cd13aa7c8a986475c461410c.

Exhibit 7

TITLE	TYPE	SEVERITY	LOCATION
Duplicated codes	Optimization	Informational	GOFETHPOOL.sol, GOFYFIIPool.sol, GOFUSDTPool.sol, GOFLINKPool.sol, GOFHTPool.sol, GOFGolffPool.sol: L715-L716,L721-722

[INFORMATIONAL] Description:

There are below duplicated codes in the if/else statements starting from L715.

Both “if” and “else” cases will execute same code snippets:

gof.mint(address(this),reward);

emit RewardAdded(reward);

So they can be moved out of the if/else and combine to one.

Recommendations:

We recommend to move the duplicated codes out of the code block and they can be replaced by one.

(Golff - resolved) The issue is addressed in commit 658c6c4ed350320a692727b7957f19ea5635cc6c.

Exhibit 8

TITLE	TYPE	SEVERITY	LOCATION
Security Vulnerability of the pool	Optimization	Major	GOFETHPOOL.sol, GOFYFIIPool.sol, GOFUSDTPool.sol, GOFLINKPool.sol, GOFHTPool.sol, GOFGolffPool.sol: L718

[MAJOR]Description:

Since the starttime is 'utc+8 2020 07-28 0:00:00'.

The start time has been passed.

When the contract is deployed, it will already pass several periods. It looks not reasonable.

So we assume the codes will be changed before deployment with a new “starttime” later than the current time.

Then there could be a case of vulnerability in the function "notifyRewardAmount".

The “rewardRate” could be maliciously manipulated. If the “notifyRewardAmount” is called before the “startTime” more than one time, the “rewardRate” will be overwritten by the last value. The GOF token would be locked in the pool forever.

Recommendations:

We recommend doing a sum of the reward before the “startTime”, and calculating the “rewardRate” correctly.

(Golff - confirmed) They will avoid to trigger this function twice before the pool startTime.

Exhibit 9

TITLE	TYPE	SEVERITY	LOCATION
Misleading Error Message	Optimization	Informational	GOF.sol:L250,L264,L302,L320

[INFORMATIONAL] Description:

The error message is not well expressed literally in several palaces.

Recommendations:

We recommend changing it like below example:

```
require(msg.sender == governance, "Golff-Token: You are not the governance");
```

(Golff - resolved) The issue is addressed in commit

2f2c47cfea5e22efc5f82d914152de5852593182.

Exhibit 10

TITLE	TYPE	SEVERITY	LOCATION
Gas consumption	Optimization	Informational	GOF.sol L38,L41,L45,L48,L52,L57,L61,L130,L133,L136,L263,L272,L280 GOFETHPOOL.sol, GOFYFIIPool.sol, GOFUSDTPool.sol, GOFLINKPool.sol, GOFHTPool.sol, GOFGolffPool.sol:L686, L690

[INFORMATIONAL] Description:

“public” functions that are never called by the contract should be declared “external” to save gas.

Recommendations:

Use the external attribute for functions never called from the contract.

(Golff - resolved) The issue is addressed in commit 2f2c47cfea5e22efc5f82d914152de5852593182.