



# 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

## 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 Teller. 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	<a href="https://github.com/golff-protocol/golff-pool">https://github.com/golff-protocol/golff-pool</a>
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 68d379beb43762e49de69f739ddcc3c8564fde9

Source Code SHA-256 Checksum

**GOFETHPOOL.sol** hash

3ce942635dfff7d838e33f368c4e044137c0fa7e2cbc2eb1cbf06a5fa179d7c1

**GOLFFPool.sol** hash

cfa9da273aa7b612fa4bf8f980ade24d85eec6985946f5f9be52c8ad58414b38

**GOFLINKPool.sol** hash

1a84aee05120b5625499312f4589a30a472bb3f599e2fbe5f0e84072b5ecf417

**GOFYFIIPool.sol** hash

dcxaaaddfe2c940a067d740dc3f8ae224aa641d490ae66f564ef02e48c1846b3

**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 68d379beb43762e49de69f739dddc3c8564fde9

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**.