GETTY/IO

Smart Contract Audit for:

Klever Finance Token (KFI)

Version 0.1: March 30th, 2021

About US

Getty/IO is an innovative remote IT consulting firm that combines software and domain expertise to build auto scalable and high performance web and mobile products. We specialise in modern Javascript technologies, lean and highly scalable backends and blockchain technologies, building outstanding products for our customers world-wide.

Born in South America, GETTY has become the largest remote development firm in the region. We are a global company helping startups and enterprises from all around the world scale their development teams by providing them the top remote developers and consultants:

- Blockchain, Big Data & Data Science
- Smart Contract Development, Review & Audit

Our professionals are able to integrate to our Customer's teams and add value from day one. Agile methodologies, experience helping dozens of teams building products and constant coaching and mentoring ensure all our engagements help our customers become successful.

In our hands your entire process is safe, without hassle and as seamless as it can be. With the collaboration of our team of experts, you can expect to achieve much more. We are present in the United States, Canada, Brazil.

Getty/IO Inc. 2019
2711 Centerville Road, STE 400,
Wilmington, Delaware, DE, 19808

Abstract

Getty/IO have been appointed by the Klever Team to carry out the audit of Klever Finance token smart contracts. **Klever** is a decentralized p2p and self-custody wallet and **KFI** is the Klever blockchain governance token, to be used in Klever's blockchain parameters and kApp upgrades. The security audit focused on verifying that the token contract fully adheres to the TRC20 token standard. During the security audit performed by Getty/IO Security on March 30 2021 no issues were discovered. The **KFI** TRC20 Token smart contract is fully compliant with the TRC20 token standard.

Disclaimer

Our responsibility is to express an opinion on these **KFI** TRC20 smart contracts based on our audit. We perform the audit to obtain reasonable assurance about whether the smart contract is safe, resilient, if it works in accordance with the specifications. This audit does not give any warranties on the security of the code. Security audit is not investment advice.

Methodology

Tests were conducted using the **Tron Shasta Testnet** and **Tron Mainnet** environments for the purpose of evaluating that the smart contract is safe, resilient, if it works in accordance with the specifications of TRC20 standard. All issues found will be written into a separate finding in the report.

A set of different techniques were used to achieve this objective, including:

Attack: A focused attempt to evaluate the quality, especially reliability, of a test object, attempting to force specific failures.

Black-box testing: Functional or non-functional tests, without reference to the internal structure of the component or system.

Boundary value analysis: A black-box test design technique in which test cases are designed based on threshold values.

Code coverage: The code analysis that determines which parts of the software were run (covered) by the test suite and which parts were not executed.

Complexity: Tests the degree to which a component or system has a design and / or internal structure that is difficult to understand, maintain and verify.

Component integration testing: Tests performed to identify defects in the interfaces and interaction between integrated components.

Component testing: The testing of individual software components.

Configuration control: Test the configuration management of a smart contract, which consists of evaluation, coordination, approval or disapproval, and implementation of changes to configuration items after formal establishment of their configuration.

Automated Audit: An automated audit using open source tools.

Manual Audit: A manual audit on smart contract code with the aims of finding security related issues, as well as flagging bad practices or inefficient designs.

NOTE: This report is subject to updates to reflect issues that can be found during new audits tests or network updates.

Getty/IO Inc. 2019
2711 Centerville Road, STE 400,
Wilmington, Delaware, DE, 19808

Audit Dashboard

Project	KFI TRC20 Token
Auditors	Bruno Campos Diogenes Ianakiara David Ianakiara Marlon Gomes
Github	Link
File Name	kfi-trc20.tar.gz
File MD5	adb93a0fa3502eaab6af4061a9b4e889
Languages	Solidity, JavaScript
Networks	Tron Testnet / Tron Mainnet
Date	2021-03-30 to 2021-03-30

Issues Found

	Minor	Medium	Major	Critical
Open	**0**	**0**	**0**	**0**
Resolved	**0**	**0**	**0**	**0**

KFI Project Overview

The **KFI** TRC20 Token smart contract implements an TRC20 token built on top of the OpenZeppelin ERC20.sol smart contract implementing pausable, detailed, capped, burnable and ownable extensions.

Token Info

Token standard(s)	TRC20
Name	Klever
Symbol	KFI
Decimals	6
Max total supply	150,000
Capped	Yes
Pausable	Yes
Mintable	Yes
Burnable	Yes

Smart Contract Audit Goals

We perform the audit to obtain reasonable assurance about whether the smart contract is safe, resilient, if it works in accordance with the specifications. The audit goal is to verify and analyze the smart contracts in four categories:

Smart Contract Security

A complete review to identify security vulnerabilities related issues within each **KFI** contract.

Smart Contract Code Quality

A complete review of the smart contract tests, and execution environments.

Smart Contract Correctness

A complete review of the smart contract source code and how it operates. Verify implementation with white paper.

Smart Contract Software Architecture

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

Scope

The audit focus was on the projects files, and test suite found in the following directories:

File	MD5
contracts/utils/SafeMath.sol	7c57767191e532699376a46e532d1727
contracts/token/kfi.sol	2b1b894d8cfe39d02307ade0c4a38f50

contracts/erc20/ERC20Mintable.sol	2d5366d1bd7846b96a49bf9590c21bcf
contracts/erc20/ERC20Detailed.sol	ce0a920943c4ac2b070ec6dfec7c8aa9
contracts/erc20/IERC20.sol	45059c3f3e44f1927a5e8f7ab454030f
contracts/erc20/ERC20.sol	75af35bf2d6bc08f63289e046cc70f6b
contracts/erc20/ERC20Pausable.sol	0efe73fac50984ec6477c6a2806cb10d
contracts/erc20/ERC20Capped.sol	8f5caa9d5ce7abeee84912f0aa5cb4ff
contracts/erc20/Pausable.sol	f401c45120aad92d361db427d574dd66
contracts/erc20/ERC20Burnable.sol	2f8aac736190e110e642c7d67df0a9d1
contracts/roles/Roles.sol	5fc00f2e2a46df9d476b6141c516b49a
contracts/roles/Ownable.sol	dc8346f3c8220d3e6adb76ddf3bb2f4b
contracts/roles/MinterRole.sol	f2ab88a8857188ef129b6923de4190f4
contracts/roles/PauserRole.sol	d110802043c68df9aeb48b67a8e61be0

Issue Overview

Getty/IO did not find any issues.

Appendix

Severity

1 - Minor

Minor issues are generally subjective in nature or potentially deal with topics like "best practices" or "readability". Minor issues will in general not indicate an actual problem or bug in code.

The maintainers should use their own judgment as to whether addressing these issues improves the codebase.

2 - Medium

Medium issues are generally objective in nature but do not represent actual bugs or security problems.

These issues should be addressed unless there is a clear reason not to.

3 - Major

Major issues will be things like bugs or security vulnerabilities. These issues may not be directly exploitable or may require a certain condition to arise in order to be exploited.

Left unaddressed, these issues are highly likely to cause problems with the operation of the contract or to lead to a situation which allows the system to be exploited in some way.