

Audit Report August, 2024







For





Table of Content

Executive Summary 02	<u>)</u>
Number of Security Issues per Severity	3
Checked Vulnerabilities	1
Techniques and Methods 05	5
Types of Severity 06	5
Types of Issues	D
LicenseFlow Contract Note	7
Medium Severity Issues 07	7
Medium Severity Issues071. Using payable.transfer might be problematic07	
	7
1. Using payable.transfer might be problematic 07	7
 Using payable.transfer might be problematic Possible loss of funds, transfer functions can silently fail 	7 3
1. Using payable.transfer might be problematic 07 2. Possible loss of funds, transfer functions can silently fail 08 Functional Tests Cases	7 33 99



Executive Summary

Project Name LicenseFlow

Overview LicenseFlow is a licensing platform for products.

Timeline 13th August 2024 - 21st August 2024

Updated Code Received NA

Second Review NA

Method Manual Review, Functional Testing, Automated Testing, etc. All the

raised flags were manually reviewed and re-tested to identify any

false positives.

Audit Scope The scope of this audit was to analyse the LicenseFlow for quality,

security, and correctness.

Source Code https://github.com/OofOne-SE/license-flow/commit/

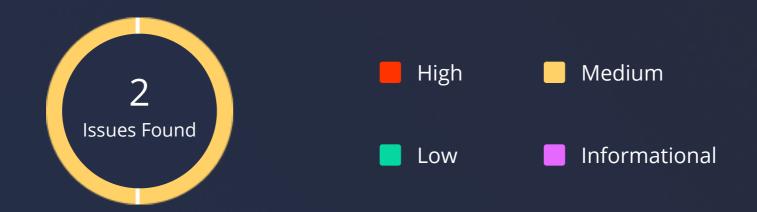
<u>fe749b21b78115e1dcae888702d21ef3b753bc5f</u>

Contracts In-Scope LicenseFlow.sol

Branch Main

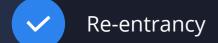
Fixed In NA

Number of Security Issues per Severity



	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	2	0	0

Checked Vulnerabilities



✓ Timestamp Dependence

Gas Limit and Loops

✓ DoS with Block Gas Limit

Transaction-Ordering Dependence

✓ Use of tx.origin

Exception disorder

Gasless send

✓ Balance equality

✓ Byte array

Transfer forwards all gas

ERC20 API violation

Compiler version not fixed

Redundant fallback function

Send instead of transfer

Style guide violation

Unchecked external call

Unchecked math

Unsafe type inference

Implicit visibility level



LicenseFlow - Audit Report

Techniques and Methods

Throughout the audit of smart contracts, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behavior.
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.
- Implementation of ERC's standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods, and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behavior of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Hardhat, Foundry.



LicenseFlow - Audit Report

Types of Severity

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

LicenseFlow contract note:

The contract is used for licensing the products. Here a single promocode can be used many times. Also a user address can use the same promo code as many times.

Medium Severity Issues

1. Using payable.transfer might be problematic

Path

LicenseFlow.sol

Function

acquireLicense(), WithdrawEthFunds()

Description

The usage of transfer so send msg.value is not recommended due to its strict dependency upon gas i.e. 2300 gas.

If gas costs are subject to change, then smart contracts can't depend on any particular gas costs.

Moreover, the function might fail mid-execution as it requires more than 2300 gas.

Recommendation

Switch to call() instead.

Status

Resolved

2. Possible loss of funds, transfer functions can silently fail

Path

LicenseFlow.sol

Function

acquireLicense()

Description

The issue here is with the use of unsafe **transfer()** function. The **ERC20.transfer()** function return a boolean value indicating success. This parameter needs to be checked for success. Some tokens do not revert if the transfer failed but return false instead.

Some tokens like USDT don't correctly implement the EIP20 standard and their transfer() function return void instead of a success boolean. Calling these functions with the correct EIP20 function signatures will always revert.

Recommendation

Use safetransfer() and safetransferfrom() instead of transfer and transferFrom.

Status

Resolved



Functional Tests Cases

- User can use the promocode many times
- Same promocode can be used many times
- Discounted prices are working correctly

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

LicenseFlow - Audit Report

Closing Summary

In this report, we have considered the security of the LicenseFlow codebase. We performed our audit according to the procedure described above.

Two Medium Severity Issues Found During the Audit, In the End, Both Issues are Fixed By LicenseFlow Team.

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in LicenseFlow smart contracts. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of LicenseFlow smart contracts. One audit is not enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your smart contracts for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services. It is the responsibility of the LicenseFlow to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

About QuillAudits

QuillAudits is a leading name in Web3 security, offering top-notch solutions to safeguard projects across DeFi, GameFi, NFT gaming, and all blockchain layers. With six years of expertise, we've secured over 1000 projects globally, averting over \$30 billion in losses. Our specialists rigorously audit smart contracts and ensure DApp safety on major platforms like Ethereum, BSC, Arbitrum, Algorand, Tron, Polygon, Polkadot, Fantom, NEAR, Solana, and others, guaranteeing your project's security with cutting-edge practices.



1000+ Audits Completed



\$30BSecured



1M+Lines of Code Audited



Follow Our Journey



















Audit Report August, 2024

For







- Canada, India, Singapore, UAE, UK
- www.quillaudits.com
- audits@quillhash.com