BLOCKAPEX

SMART CONTRACT SECURITY

V 1.0

DATE: 24th APR, 2024

PREPARED FOR: LIGHTLINK LUMINARY

SECURITY REPORT

BLOCKAPEX VERIFIED

About BlockApex

Founded in early 2021, is a security-first blockchain consulting firm. We offer services in a wide range of areas including Audits for Smart Contracts, Blockchain Protocols, Tokenomics along with Invariant development (i.e., test-suite) and Decentralized Application Penetration Testing. With a dedicated team of over 40+ experts dispersed globally, BlockApex has contributed to enhancing the security of essential software components utilized by many users worldwide, including vital systems and technologies.

BlockApex has a focus on blockchain security, maintaining an expertise hub to navigate this dynamic field. We actively contribute to security research and openly share our findings with the community. Our work is available for review at our public repository, showcasing audit reports and insights into our innovative practices.

To stay informed about BlockApex's latest developments, breakthroughs, and services, we invite you to follow us on Twitter and explore our GitHub. For direct inquiries, partnership opportunities, or to learn more about how BlockApex can assist your organization in achieving its security objectives, please visit our Contact page at our website , or reach out to us via email at hello@blockapex.io.

Contents

1 Executive Summary **4**

1.1 Scope . . . 5

1.1.1 In Scope . . . 5

1.1.2 Out of Scope . . . 5

1.2 Methodology . . . 5

1.3 Code Coverage . . . 6

1.4 Questions for Security Assessment . . . 6

1.5 Status Descriptions . . . 7

1.6 Summary of Findings Identified . . . 8

2 Findings and Risk Analysis **9**

2.1 Return calls not checked for transfer & transferFrom function . . . 9

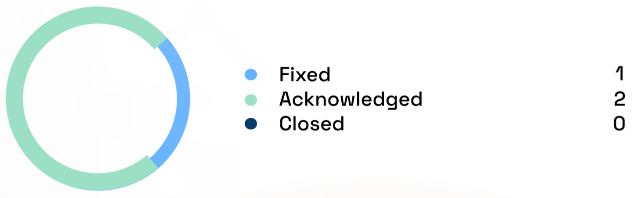
2.2 Insufficient signature validation for multiple domains . . . 10

2.3 Absence of an event in withdraw function . . . 11

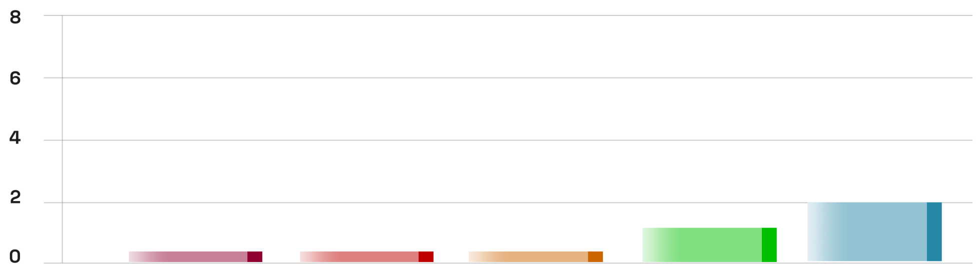
1 Executive Summary

The audit was conducted on the LightLink Luminary smart contract, undertaken by an individual auditor. This comprehensive review employed a meticulous manual code analysis approach, focusing exclusively on a line-by-line examination of the contract's code to identify potential vulnerabilities, coding best practices deviations, and areas for optimization

Developer Response



Issues Overview



• Critical	• High	• Medium	• Low	• Info
Critical vulnerabilities are too easy to exploit and can lead to damages/loss in assets or manipulations.	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution.	Medium-level vulnerabilities are equally imperative to fix but they tend to have minimal impact on asset loss or data manipulations.	Lowest-level vulnerabilities, informational errors, violating code styles/ practices usually can't affect smart contract execution; hence they can be ignored.	Concerns that have no direct negative impact. These may help improve coding practices.

1.1 Scope

1.1.1 In Scope

The project in scope involved different components mentioned below.

LL Token - This is an ERC-20 implementation of LightLink token.

LL Claim - This is claim contract that allows users to claim their tokens in different phases aided by off-chain infrastructure for signature generation.

LuminaKey & Luminary - These inherit the implementation of ERC-5725, Transferable Vesting NFT which is an interface for transferable vesting NFTs which release underlying tokens over time. Luminary & LuminaKey contracts have simple vesting strategy implemented which allows users to claim tokens after fixed amount of time.

Contracts in Scope:

All files under the folder Contracts/*

Initial Commit Hash: 2e37204de5706a9275256920a77b7134591df3bd

Final Commit Hash: 2cf9bd37f023c782bd5a339a049e2ced805052b5

1.1.2 Out of Scope

All features or functionalities not delineated within the "In Scope" section of this document shall be deemed outside the review of this audit. This exclusion particularly applies to the backend operations of the Luminary & LuminaKey contracts associated with the platform & any other external libraries

1.2 Methodology

The audit of the LightLink Luminary smart contract was conducted over the course of two days, utilizing a straightforward, manual code review approach by one auditor. The methodology began with a reconnaissance phase to gain an initial understanding of the contract's structure and intended functionalities. Following this, the auditor engaged in a detailed, line-by-line examination of the code. This manual review process focused on identifying logical flaws, security vulnerabilities, and opportunities for optimization, while ensuring adherence to Solidity best practices, security design patterns, and coding standards for clarity and efficiency.

1.3 Code Coverage

This section provides an overview of the analysis coverage of the review, as determined by our high-level engagement goals. Our approaches included the following:

File: src/ERC5725.sol

- **Lines Coverage:** 86.67% (26/30)
- **Statements Coverage:** 85.42% (41/48)
- **Branches Coverage:** 66.67% (8/12)
- **Functions Coverage:** 93.33% (14/15)

File: src/LLClaim.sol

- **Lines Coverage:** 84.62% (11/13)
- **Statements Coverage:** 87.50% (14/16)
- **Branches Coverage:** 83.33% (5/6)
- **Functions Coverage:** 71.43% (5/7)

File: src/VestingNFT.sol

- **Lines Coverage:** 95.45% (21/22)
- **Statements Coverage:** 96.67% (29/30)
- **Branches Coverage:** 100.00% (4/4)
- **Functions Coverage:** 92.31% (12/13)

Overall the testcases seems to cover most of the functionality, It is however recommended to introduce more comprehensive test-cases to pin point undefined behaviour and edge-cases.

1.4 Questions for Security Assessment

The engagement was scoped to provide a security assessment of the LL - Luminary smart contract. Specifically, we sought to answer the following non-exhaustive list of questions:

1. There any potential signature manipulation attack possible ?
2. Can a scenario be created leveraging block.timestamp for a longer attack chain ?
3. Is there any reentrancy vulnerability present in the contract ?
4. Does the vesting strategy have any edge cases ?
5. Is there any issue related to the approver claiming tokens on behalf of the user ?
6. Is there any possibility of double claims ?

1.5 Status Descriptions

Acknowledged: The issue has been recognized and is under review. It indicates that the relevant team is aware of the problem and is actively considering the next steps or solutions.

Fixed: The issue has been addressed and resolved. Necessary actions or corrections have been implemented to eliminate the vulnerability or problem.

Closed: This status signifies that the issue has been thoroughly evaluated and acknowledged by the development team. While no immediate action is being taken.

1.6 Summary of Findings Identified

S.No	Severity	Findings	Status
#1	LOW	Return calls not checked for transfer & transferFrom function	FIXED
#2	INFO	Insufficient signature validation for multiple domains	ACKNOWLEDGED
#3	INFO	Absence of an event in withdraw function	ACKNOWLEDGED

2 Findings and Risk Analysis

2.1 Return calls not checked for transfer & transferFrom function

Severity: Low

Status: Fixed

Location :

1. `claim` function L#21, `ERC5725`
2. `withdraw` function L#50, `LLClaim`
3. `mint` function L#63, `VestingNFT`

Description The function `claim`, `withdraw` & `mint` in the contracts mentioned below serves as a helper function to transfer assets to a target address in one function call. Per the ERC-20 standard, calls to the transfer function return a Boolean indicating whether the call was successful, and developers must not assume that false is never returned and that the token contract will revert instead. However, the return value of the call to transfer is not checked and given the history of differing ERC-20 implementations in the Ethereum ecosystem, validating the outcome of these function calls is strongly recommended.

Recommendation Add a check to verify that the transfer call in these was successful

2.2 Insufficient signature validation for multiple domains

Severity: Info

Status: Acknowledged

Location :

1. `_validateSignature function` L#54, `LLClaim`
2. `_validateSignature` function L#122, `VestingNFT`

Description These functions provide a way to validate an off-chain signature created by the signer to use on-chain for weather `minting` or `claiming` vesting ERC20 tokens. The implementation uses a separate function for storing salt makes is used to verify the signature which solves the problem for the same chain. If the `VestingNFT` contract is intended to be used on multi-chain, the signatures present on one chain will be automatically valid on another chain which might lead to unexpected scenarios. Therefore it is recommended to introduce a systematic scheme which utilizes `chainID`, `contractAddress` & `domain` specific to the contract to generate & validate signatures

Recommendation Introduce EIP-712 mechanism to address separation of domains across multiple contracts on same network & on different chains. See (EIP-712). Implementing this will remove the need for `salt` as an external function

2.3 Absence of an event in withdraw function

Severity: Info

Status: Acknowledged

Location :

1. `withdraw` function L#49, `LLClaim`

Description The withdraw function lack the necessary event emission when the withdraw function is called. This could be necessary by minting records off-chain for tracking purposes

Recommendation Add an event which is emitted when withdraw function is called.