LOKR

MINTR

Smart Contract Audit
Final Report

December 26, 2022

Introduction 3
1. About LOKR 3
2. About ImmuneBytes 3

Documentation Details 3

Audit Process & Methodology 4

Audit Details 4

Audit Goals 5

Security Level Reference 5

Audit Summary 6

Finding 7

Audit Changes 8
File Changes 8

Additional Changes 8

Concluding Remarks 9

Disclaimer 9

Introduction

1. About LOKR

Lokr provides projects and individuals with a full suite of tools to manage their token economies. Whether it be token creation, distribution, creating cross-chain bridging solutions, or trustless escrows, Lokr has you covered. Lokr lowers the barrier of entry into the crypto space, making blockchain accessible to everyone through simple plug-and-play cross-chain solutions.

Mintr:

Ethereum, Binance Smart Chain and Polygon, Mintr will eventually expand to include all EVM compatible chains. With it's intuitive design, Mintr aims to lower the barrier for entry for users entering the space. Creating contracts is now easier than ever as no coding experience is necessary and, if required, users can request an audit report for their token right off the bat, saving both time and money. Mintr's dashboard publicly displays all minted tokens and their capabilities, bringing another layer of transparency to the community.

Visit https://www.lokr.io/ to know more about it.

2. About ImmuneBytes

ImmuneBytes is a security start-up that provides professional services in the blockchain space. The team has hands-on experience conducting smart contract audits, penetration testing, and security consulting. ImmuneBytes's security auditors have worked on various A-league projects and understand DeFi projects like AAVE, Compound, 0x Protocol, Uniswap, and dydx.

The team has secured 205+ blockchain projects by providing security services on different frameworks. The ImmuneBytes team helps start-ups with detailed system analysis, ensuring security and managing the overall project.

Visit http://immunebytes.com/ to learn more about the services.

Documentation Details

The LOKR team has provided the following doc for audit:

1. https://immunebytes.notion.site/Standard-Contract-LOKR-mintr-eeccb0e648e440289ce7e6fd175c959e

Audit Process & Methodology

ImmuneBytes team has performed thorough testing of the project, starting with analyzing the code design patterns in which we reviewed the smart contract architecture to ensure it is structured and safe use of third-party smart contracts and libraries.

Our team then performed a formal line-by-line inspection of the Smart Contract to find potential issues like Signature Replay Attacks, Unchecked External Calls, External Contract Referencing, Variable Shadowing, Race conditions, Transaction-ordering dependence, timestamp dependence, DoS attacks, and others.

In the Unit testing phase, we run unit tests written by the developer to verify the functions work as intended. In Automated Testing, we tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

The code was audited by a team of independent auditors, including -

1. Structural analysis of the smart contract is checked and verified.
2. An extensive automated testing of all the contracts under scope is conducted.
3. Line-by-line Manual Code review is conducted to evaluate, analyze and identify the potential security risks in the contract.
4. Evaluation of the contract's intended behavior and the documentation shared is imperative to verify the contract behaves as expected.
5. For complex and heavy contracts, adequate integration testing is conducted to ensure that contracts interact acceptably.
6. Storage layout verifications in the upgradeable contract are a must.
7. An important step in the audit procedure is highlighting and recommending better gas optimization techniques in the contract.

Audit Details

- Project Name: LOKR
- Languages: Solidity(Smart contract), Typescript (Unit Testing)
- Github link: https://github.com/Polkalokr/mintr-contracts/tree/deploy
- Commit hash: d4768de0c410d5d939def638224c1b0c1dbb2820
- Platforms and Tools: Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, Contract Library, Slither, SmartCheck

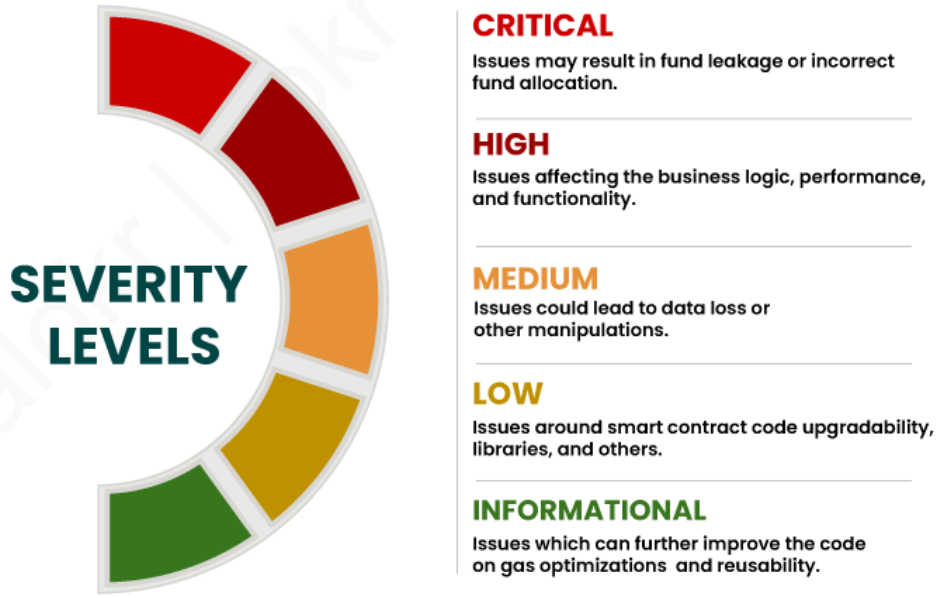
Audit Goals

The audit's focus was to verify that the smart contract system is secure, resilient, and working according to its specifications. The audit activities can be grouped into the following three categories:

1. Security: Identifying security-related issues within each contract and the system of contracts.
2. Sound Architecture: Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.
3. Code Correctness and Quality: A full review of the contract source code. The primary areas of focus include
    a. Correctness
    b. Readability
    c. Sections of code with high complexity
    d. Quantity and quality of test coverage

Security Level Reference

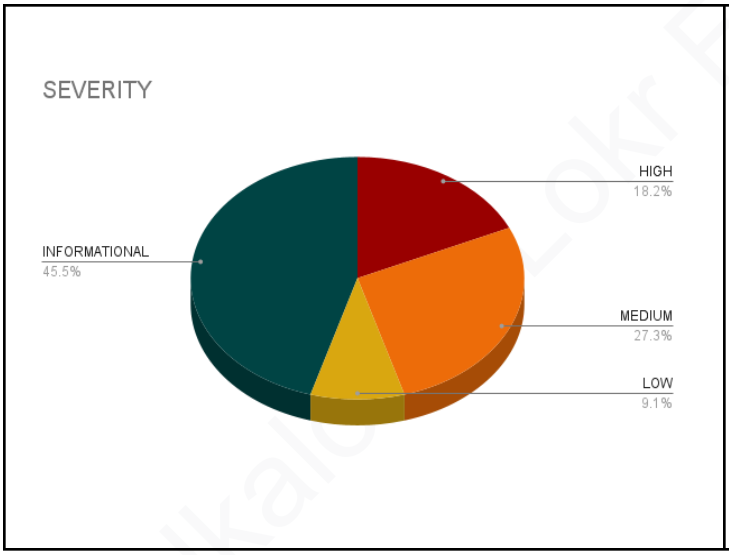
Every issue in this report were assigned a severity level from the following:

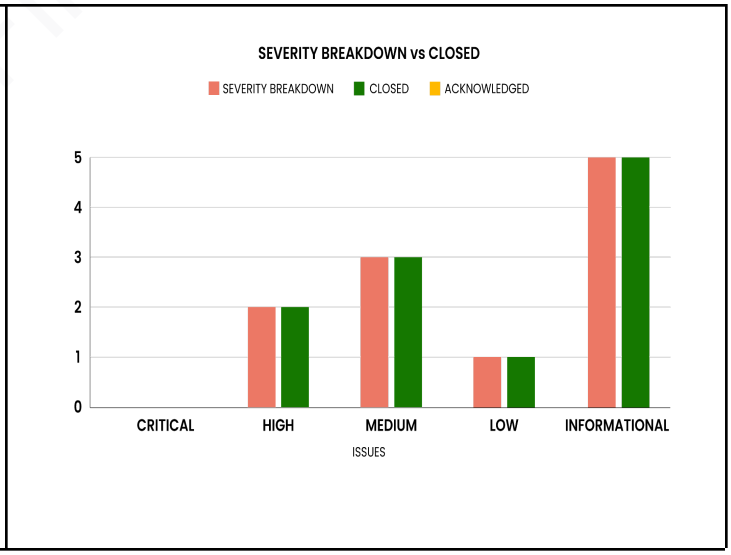


Audit Summary

Team ImmuneBytes has performed a line-by-line manual analysis and automated review of smart contracts. Smart contracts were analyzed mainly for common contract vulnerabilities, exploits, and manipulation hacks. According to the audit:

Issues	Critical	High	Medium	Low
Open	-	-	-	-
Closed	-	2	3	1
Acknowledged	-	-	-	-





Finding

S.No	Findings	Risk	Status
1	TxFeeFeatureV3.sol : Invalid Updation of caller's currentAllowance token amount in transferFrom function	High	Closed
2	TxFeeFeatureV3.sol, TxFeeFeatureAlt: Conditions for charging transaction fees are wrongly implemented in transfer & transferFrom function	High	Closed
3	TxFeeFeatureV3.sol, TxFeeFeatureAlt: Conditions for charging transaction fees aren't consistent	Medium	Closed
4	PausableWithWhitelistFeature.sol: Whitelisted token addresses can transact even if the contract is Paused	Medium	Closed
5	PausableWithWhitelistFeature.sol: _checkWhitelist function shall return true even if only one address is whitelisted	Medium	Closed
6	TxFeeFeatureV3.sol, TxFeeFeatureAlt.sol, TxFeeFeature.sol: Contradictory	Low	Closed
7	TxFeeFeatureV3.sol, TxFeeFeatureAlt.sol, TxFeeFeature.sol: Variables read from storage multiple times lead to more gas consumption	Informatory	Closed
8	PausableWithWhitelistFeature.sol: Inadequate Error message found in require statement	Informatory	Closed
9	TxFeeFeatureV3.sol, TxFeeFeatureAlt.sol, TxFeeFeature.sol: Absence of Error messages in Require Statements	Informatory	Closed
10	TxFeeFeatureV3.sol, TxFeeFeatureAlt.sol, TxFeeFeature.sol: Inadequate natspec statements assigned to contract	Informatory	Closed
11	TxFeeFeatureAlt.sol: Coding Style Issues in the Contract	Informatory	Closed

Audit Changes

The LOKR team has implemented the recommendations based on the auditor's initial finding as under:

File Changes

1. **contracts/features/TxFeeFeatureV3.sol(Issue # 1,2,3,6,7,9 & 10)**
    - Inherited contract name "ITxFeeFeatureAlt" replace with "ITxFeeFeatureV3" on line # 13.
    - Change require statement on line # 27-48.
    - Replace contract name from "ITxFeeFeatureAlt" to "ITxFeeFeatureV3" on line # 58 & 84.
    - Change check condition on line # 68.
    - Replace variable name StorageSlotUpgradeable.getUint256Slot(TX_FEE_SLOT).value with txfee on line # 70 & 96.
    - Update transfer function on line # 75 & 101.
    - Change condition from line # 90-95.
    - Change approve function on line # 107.
    - Create new function on line # 113-120.

2. **contracts/features/TxFeeFeatureALT.sol.(Issue # 11)**
    - Change variable name exp to EXP on line # 18.
    - Change require statement on line # 22-31.
    - Change check condition on line # 47.
    - Update variable name on line # 48 & 71.
    - Change check condition on line # 67-70.
    - Create new function on line # 85-91.

3. **contracts/features/PausableWithWhitelistFeature.sol.(Issue # 4, 5 & 8)**
    - Update return statement on line # 74.

4. **contracts/features/TxFeeFeature.sol(issue # 6,7 9 & 10)**
    - Change variable name exp to EXP on line # 15.
    - Change require statement on line # 19-28.
    - Update variable name on line # 44.

Additional Changes

No additional changes were made during the code refactor.

Concluding Remarks

While conducting the audits of the LOKR smart contracts, it was observed that the contracts contain High, Medium, and Low severity issues.

Note:
The LOKR team has fixed the issues based on the auditor's recommendation.

Disclaimer

ImmuneBytes's audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process; therefore, running a bug bounty program complementing this audit is strongly recommended.

Our team does not endorse the LOKR platform or its product, nor this audit is investment advice.

Notes:

- Please make sure contracts deployed on the mainnet are the ones audited.
- Check for code refactoring by the team on critical issues.