SMART CONTRACTS REVIEW

zokyo

May 30th 2024 | v. 1.0

Security Audit Score

PASS

Zokyo Security has concluded that this smart contract passed a security audit.

ZOKYO AUDIT SCORING DEVVE

1. Severity of Issues:
    - Critical: Direct, immediate risks to funds or the integrity of the contract. Typically, these would have a very high weight.
    - High: Important issues that can compromise the contract in certain scenarios.
    - Medium: Issues that might not pose immediate threats but represent significant deviations from best practices.
    - Low: Smaller issues that might not pose security risks but are still noteworthy.
    - Informational: Generally, observations or suggestions that don't point to vulnerabilities but can be improvements or best practices.
2. Test Coverage: The percentage of the codebase that's covered by tests. High test coverage often suggests thorough testing practices and can increase the score.
3. Code Quality: This is more subjective, but contracts that follow best practices, are well-commented, and show good organization might receive higher scores.
4. Documentation: Comprehensive and clear documentation might improve the score, as it shows thoroughness.
5. Consistency: Consistency in coding patterns, naming, etc., can also factor into the score.
6. Response to Identified Issues: Some audits might consider how quickly and effectively the team responds to identified issues.

HYPOTHETICAL SCORING CALCULATION:

Let's assume each issue has a weight:
- Critical: -30 points
- High: -20 points
- Medium: -10 points
- Low: -5 points
- Informational: -1 point

Starting with a perfect score of 100:
- 0 Critical issues: 0 points deducted
- 0 High issues: 0 points deducted
- 0 Medium issues: 0 points deducted
- 1 Low issue: 1 acknowledged = - 3 points deducted
- 3 Informational issues: 3 acknowledged = 0 points deducted

Thus, 100 - 3 = 97

TECHNICAL SUMMARY

This document outlines the overall security of the Devve smart contract/s evaluated by the Zokyo Security team.

The scope of this audit was to analyze and document the Devve smart contract/s codebase for quality, security, and correctness.

Contract Status

There were 0 critical issues found during the review. (See Complete Analysis)

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract/s but rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that can withstand the Ethereum network's fast-paced and rapidly changing environment, we recommend that the Devve team put in place a bug bounty program to encourage further active analysis of the smart contract/s.

Table of Contents

Auditing Strategy and Techniques Applied 5

Executive Summary 7

Structure and Organization of the Document 8

Complete Analysis 9

AUDITING STRATEGY AND TECHNIQUES APPLIED

The source code of the smart contract was taken from the Devve repository:
Repo: https://etherscan.io/address/0×1d64231e179f074baa3f22d44a6ca3a2670b2709#code

Within the scope of this audit, the team of auditors reviewed the following contract(s):

- Fias.sol

During the audit, Zokyo Security ensured that the contract:

- Implements and adheres to the existing standards appropriately and effectively;
- The documentation and code comments match the logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices, efficiently using resources without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the most recent vulnerabilities;
- Meets best practices in code readability, etc.

Zokyo Security has followed best practices and industry-standard techniques to verify the implementation of Devve smart contract/s. To do so, the code was reviewed line by line by our smart contract developers, who documented even minor issues as they were discovered. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

01 Due diligence in assessing the overall code quality of the codebase.

02 Cross-comparison with other, similar smart contract/s by industry leaders.

03 Thorough manual review of the codebase line by line.

Executive Summary

The Fias.sol smart contract is an ERC20Upgradeable smart contract implementation. The contract defines two specific addresses, each assigned a total of 175,000,000 mintable tokens, making the total supply 350,000,000 tokens. The smart contract implements two mint functions and a multiMint function, allowing minters to mint tokens to different addresses. Additionally, the contract implements a multiTransfer function that allows sending several amounts of tokens to multiple different addresses at once.

STRUCTURE AND ORGANIZATION OF THE DOCUMENT

For the ease of navigation, the following sections are arranged from the most to the least critical ones. Issues are tagged as “Resolved” or “Unresolved” or “Acknowledged” depending on whether they have been fixed or addressed. Acknowledged means that the issue was sent to the Devve team and the Devve team is aware of it, but they have chosen to not solve it. The issues that are tagged as “Verified” contain unclear or suspicious functionality that either needs explanation from the Client or remains disregarded by the Client. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

FINDINGS SUMMARY

#	Title	Risk	Status
1	Hardcoded addresses are used for crucial roles	Low	Acknowledged
2	Unnecessary checks and initializations	Informational	Acknowledged
3	Unused admin address	Informational	Acknowledged
4	The `mint` function relies on centralized entities in order to function properly	Informational	Acknowledged

LOW-1 | ACKNOWLEDGED

Hardcoded addresses are used for crucial roles

The `Fias.sol` contract implements 3 crucial roles, `FOREVVER_MINTER`, `LITCRAFT_MINTER`, `CONTRACT_ADMIN`. These addresses are hardcoded, this can lead to a scenario where the protocol is deployed to the same or another chain in the future where/when one of these addresses is no longer safe or admin lose their private keys.

Recommendation:
Set the addresses in the `initialize()` function instead of hardcoding them. Take into account that the transaction can be front-run if the addresses are set in the `initialize()` function.

INFORMATIONAL-1 | ACKNOWLEDGED

Unnecessary checks and initializations

There are some unnecessary or redundant checks within the `Fias.sol` contract:

- Initialization to default value in `initialize()` function:

```
function initialize() public initializer {
        __ERC20_init("Fias", "FIAS");
        _explicitMinter[LITCRAFT_MINTER] = true;
        _explicitMinter[FOREVVER_MINTER] = true;
        _minterLimit[LITCRAFT_MINTER] = LITCRAFT_LIMIT;
        _minterLimit[FOREVVER_MINTER] = FOREVVER_LIMIT;
        _mintedAmount[LITCRAFT_MINTER] = 0; // @audit not needed
        _mintedAmount[FOREVVER_MINTER] = 0; // @audit not needed
    }
```

- Redundant `_explicitMinter[msg.sender]` check in `multiMint()` function:

```
function multiMint(address[] memory recipients, uint256[] memory amounts) public {
        require(_explicitMinter[msg.sender], "Caller does not have the minter role"); //
@audit not needed
        require(recipients.length > 0, "No mint recipients.");
        require(recipients.length == amounts.length, "Recipient/Amount mismatch.");
        for (uint256 i = 0; i < recipients.length; i++) {
            mint(recipients[i], amounts[i]);
        }
    }
```

Recommendation:
Remove the unnecessary checks and initializations in order to save gas.

INFORMATIONAL-2 | ACKNOWLEDGED

Unused admin address

The `Fias.sol` contract contains a `CONTRACT_ADMIN` address which is not used in any part of the code.

Recommendation:
Consider removing this variable to save gas.

INFORMATIONAL-3 | ACKNOWLEDGED

The `mint` function relies on centralized entities in order to function properly

The `mint` function is only available for 2 defined addresses, `LITCRAFT_MINTER` and `FOREVVER_MINTER`. These are the only 2 addresses that can mint tokens. Having a centralized scenario can imply several risks as losing the control of one of the addresses and not being able to mint more tokens.

	ERC20 Token 0x1d64231e179f074baa3f22d44 a6ca3a2670b2709
Reentrancy	Pass
Access Management Hierarchy	Pass
Arithmetic Over/Under Flows	Pass
Unexpected Ether	Pass
Delegatecall	Pass
Default Public Visibility	Pass
Hidden Malicious Code	Pass
Entropy Illusion (Lack of Randomness)	Pass
External Contract Referencing	Pass
Short Address/ Parameter Attack	Pass
Unchecked CALL Return Values	Pass
Race Conditions / Front Running	Pass
General Denial Of Service (DOS)	Pass
Uninitialized Storage Pointers	Pass
Floating Points and Precision	Pass
Tx.Origin Authentication	Pass
Signatures Replay	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass

We are grateful for the opportunity to work with the Devve team.

The statements made in this document should not be interpreted as an investment or legal advice, nor should its authors be held accountable for the decisions made based on them.

Zokyo Security recommends the Devve team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo