



# # Competitive Security Assessment

RevoxAI

Jul 3rd, 2024



---

Summary	3
Overview	4
Audit Scope	5
Code Assessment Findings	6
RAI-1 Use safeTransfer consistently instead of transfer	7
RAI-2 Interfaces have been deprecated	8
RAI-3 Two-step ownership transfer	9
RAI-4 Lack of Zero Address Validation	10
RAI-5 <code>setDepositMap</code> does not check for price > 0	11
RAI-6 Missing emit events	12
Disclaimer	13

## Summary

This report is prepared for the project to identify vulnerabilities and issues in the smart contract source code. A group of NDA covered experienced security experts have participated in the Secure3's Audit Contest to find vulnerabilities and optimizations. Secure3 team has participated in the contest process as well to provide extra auditing coverage and scrutiny of the finding submissions.

The comprehensive examination and auditing scope includes:

- Cross checking contract implementation against functionalities described in the documents and white paper disclosed by the project owner.
- Contract Privilege Role Review to provide more clarity on smart contract roles and privilege.
- Using static analysis tools to analyze smart contracts against common known vulnerabilities patterns.
- Verify the code base is compliant with the most up-to-date industry standards and security best practices.
- Comprehensive line-by-line manual code review of the entire codebase by industry experts.

The security assessment resulted in findings that are categorized in four severity levels: Critical, Medium, Low, Informational. For each of the findings, the report has included recommendations of fix or mitigation for security and best practices.

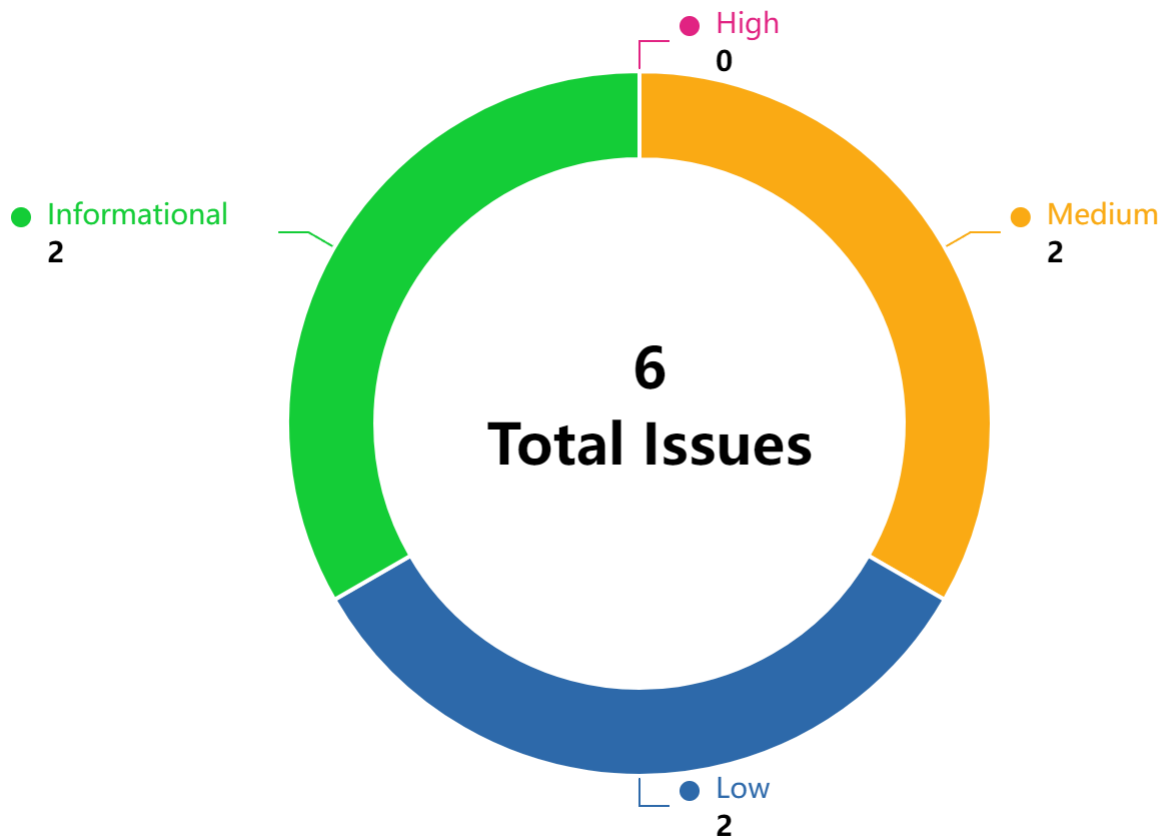
# Overview

Project Name	RevoxAI
Language	solidity
Codebase	<ul style="list-style-type: none"><li>• <a href="https://github.com/readonme/contracts_solidity">https://github.com/readonme/contracts_solidity</a></li><li>• audit version - 30937cd582fb408f073f3c112a75e4870a790a09</li><li>• final version - 57bfe117bc9363dc2bf0d305c8006e1e9a0a66b9</li></ul>
Audit Methodology	<ul style="list-style-type: none"><li>• Audit Contest</li><li>• Business Logic and Code Review</li><li>• Privileged Roles Review</li><li>• Static Analysis</li></ul>

## Audit Scope

File	SHA256 Hash
lense/LenseCore.sol	69b6c6e10a287113e605995e1f0c88ce1d2b79c979cf1ec7107b8c519f830919
curate/ReadonCurateV2.sol	9ff5f88dbdd67486fa5dd73d0ecfb845c39d3d323b7f58a36cae60bcbfc329e3

# Code Assessment Findings



ID	Name	Category	Severity	Client Response	Contributor
RAI-1	Use safeTransfer consistently instead of transfer	Code Style	Medium	Fixed	***
RAI-2	Interfaces have been deprecated	Code Style	Medium	Fixed	***
RAI-3	Two-step ownership transfer	Logical	Low	Acknowledged	***
RAI-4	Lack of Zero Address Validation	Logical	Low	Fixed	***
RAI-5	setDepositMap does not check for price > 0	Logical	Informational	Fixed	***
RAI-6	Missing emit events	Code Style	Informational	Fixed	***

# RAI-1:Use safeTransfer consistently instead of transfer

Category	Severity	Client Response	Contributor
Code Style	Medium	Fixed	***

## Code Reference

- code/lense/LenseCore.sol#L79

```
79: IERC20Upgradeable(token).transfer(
```

## Descriptionhide

\*\*\*: Some tokens (like USDT) don't correctly implement the EIP20 standard and their `transfer/transferFrom` function return void instead of a boolean.

## Recommendationhide

\*\*\*: Use call `safeTransfer` when transferring ERC20 tokens.

## Client Responsehide

client response for 0xCO2: Fixed. [https://github.com/readonme/contracts\\_solidity/commit/57bfe117bc9363dc2bf0d305c8006e1e9a0a66b9](https://github.com/readonme/contracts_solidity/commit/57bfe117bc9363dc2bf0d305c8006e1e9a0a66b9)

## RAI-2:Interfaces have been deprecated

Category	Severity	Client Response	Contributor
Code Style	Medium	Fixed	***

### Code Reference

- code/lense/LenseCore.sol#L11
- code/lense/LenseCore.sol#L12

```
11: import "@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol";
```

```
12: import "@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol";
```

### Descriptionhide

\*\*\*: The interfaces `IERC20Upgradeable` and `SafeERC20Upgradeable` have been deprecated now. It may cause potential conflicts.

### Recommendationhide

\*\*\*: Please check [here](#).

### Client Responsehide

client response for 0xCO2: Fixed. [https://github.com/readonme/contracts\\_solidity/commit/57bfe117bc9363dc2bf0d305c8006e1e9a0a66b9](https://github.com/readonme/contracts_solidity/commit/57bfe117bc9363dc2bf0d305c8006e1e9a0a66b9)



# RAI-3:Two-step ownership transfer

Category	Severity	Client Response	Contributor
Logical	Low	Acknowledged	***

## Code Reference

- code/curate/ReadonCurateV2.sol#L4
- code/curate/ReadonCurateV2.sol#L25

```
4: import "@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol";
```

```
25: __Ownable_init();
```

- code/lense/LenseCore.sol#L4
- code/lense/LenseCore.sol#L45

```
4: import "@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol";
```

```
45: __Ownable_init();
```

## Descriptionhide

\*\*\*: For example, transferring ownership to an incorrect account or to a contract that cannot interact with the permission system can result in irreversible losses.

## Recommendationhide

\*\*\*: The \_\_Ownable2Step\_init() function of Ownable2StepUpgradeable.sol should be used.

## Client Responsehide

client response for xyzqwe123: Acknowledged.

## RAI-4:Lack of Zero Address Validation

Category	Severity	Client Response	Contributor
Logical	Low	Fixed	***

### Code Reference

- code/lense/LenseCore.sol#L49-L51

```
49: function setToken(address _token) external onlyOwner {
50:     token = _token;
51: }
```

### Descriptionhide

\*\*\*: The function `setToken()` is used to set token address, however, it lacks parameter validation.

\*\*\*: Functon `setToken` lacks validation to check whether the parameter `_token` is the zero address, while functions `deposit/payForOnce/withdrawTokens` do not handle the case when the token address is zero.

\*\*\*: If the token address is set to zero, it will cause the deposit function, payForOnce function, and withdrawTokens function to fail to execute properly.

### Recommendationhide

\*\*\*: Consider that add require condition to check for zero address.

\*\*\*: Consider adding `require()` condition for zero address validation.

```
function setToken(address _token) external onlyOwner {
    require(_token != address(0x0), "_token is zero address");
    token = _token;
}
```

\*\*\*: Add a zero address check.

### Client Responsehide

client response for 0xCO2: Fixed. [https://github.com/readonme/contracts\\_solidity/commit/57bfe117bc9363dc2bf0d305c8006e1e9a0a66b9](https://github.com/readonme/contracts_solidity/commit/57bfe117bc9363dc2bf0d305c8006e1e9a0a66b9)

client response for jiayeqian: Fixed. [https://github.com/readonme/contracts\\_solidity/commit/57bfe117bc9363dc2bf0d305c8006e1e9a0a66b9](https://github.com/readonme/contracts_solidity/commit/57bfe117bc9363dc2bf0d305c8006e1e9a0a66b9)

client response for xyzqwe123: Fixed. [https://github.com/readonme/contracts\\_solidity/commit/57bfe117bc9363dc2bf0d305c8006e1e9a0a66b9](https://github.com/readonme/contracts_solidity/commit/57bfe117bc9363dc2bf0d305c8006e1e9a0a66b9)

## RAI-5: `setDepositMap` does not check for price > 0

Category	Severity	Client Response	Contributor
Logical	Informational	Fixed	***

### Code Reference

- code/lense/LenseCore.sol#L62

```
62: depositMap[tier] = price;
```

### Descriptionhide

\*\*\*: the price for the tier is not checked is not zero, however, in the `deposit` function it `require(amount > 0, "tier not exist");` and there is inconsistency

### Recommendationhide

\*\*\*: add `require(price > 0, "invalid price");`

### Client Responsehide

client response for toffee: Fixed. [https://github.com/readonme/contracts\\_solidity/commit/57bfe117bc9363dc2bf0d305c8006e1e9a0a66b9](https://github.com/readonme/contracts_solidity/commit/57bfe117bc9363dc2bf0d305c8006e1e9a0a66b9)

## RAI-6:Missing emit events

Category	Severity	Client Response	Contributor
Code Style	Informational	Fixed	***

### Code Reference

- code/lense/LenseCore.sol#L49
- code/lense/LenseCore.sol#L53
- code/lense/LenseCore.sol#L61

```
49: function setToken(address _token) external onlyOwner {
```

```
53: function setFeePrice(uint256 _feePrice) external onlyOwner {
```

```
61: function setDepositMap(uint256 tier,uint256 price) external onlyOwner {
```

### Descriptionhide

\*\*\*: Functions ( `setToken/setFeePrice/setDepositMap` ) do not emit events, so it is difficult to track changes in the values of `token/feePrice/depositMap` .

### Recommendationhide

\*\*\*: Emit an event for gloab parameter( `token/feePrice/depositMap` ) changes.

### Client Responsehide

client response for jiayeqian: Fixed. [https://github.com/readonme/contracts\\_solidity/commit/57bfe117bc9363dc2bf0d305c8006e1e9a0a66b9](https://github.com/readonme/contracts_solidity/commit/57bfe117bc9363dc2bf0d305c8006e1e9a0a66b9)

# Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Invoices, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Invoice. This report provided in connection with the services set forth in the Invoices shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Invoice. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without Secure3's prior written consent in each instance.

This report is not an "endorsement" or "disapproval" of any particular project or team. This report is not an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Secure3 to perform a security assessment. This report does not provide any warranty or guarantee of free of bug of codes analyzed, nor do they provide any indication of the technologies, business model or legal compliancy.

This report should not be used in any way to make decisions around investment or involvement with any particular project. Instead, it represents an extensive assessing process intending to help our customers increase the quality of their code and high-level consistency of implementation and business model, while reducing the risk presented by cryptographic tokens and blockchain technology.

Secure3's position on the final decisions over blockchain technologies and corresponding associated transactions is that each company and individual are responsible for their own due diligence and continuous security.

The assessment services provided by Secure3 is subject to dependencies and under continuing development. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

