F Y E O

Security Code Review
Theoriq

Theoriq

August 2024
Version 1.0

Presented by:
FYEO Inc.
PO Box 147044
Lakewood CO 80214
United States

Security Level
Public

Table of Contents

Executive Summary

Overview

Theoriq engaged FYEO Inc. to perform a Security Code Review of the updates to the Theoriq smart contract.

The assessment was conducted remotely by the FYEO Security Team. Testing took place on July 31 - August 03, 2024, and focused on the following objectives:

- To provide the customer with an assessment of their overall security posture and any risks that were discovered within the environment during the engagement.
- To provide a professional opinion on the maturity, adequacy, and efficiency of the security measures that are in place.
- To identify potential issues and include improvement recommendations based on the results of our tests.

This report summarizes the engagement, tests performed, and findings. It also contains detailed descriptions of the discovered vulnerabilities, steps the FYEO Security Team took to identify and validate each issue, as well as any applicable recommendations for remediation.

Key Findings

The following issues have been identified during the testing period. These should be prioritized for remediation to reduce the risk they pose:

- FYEO-THEORIQ-01 – Input arrays may be of different lengths
- FYEO-THEORIQ-02 – Missing reentrancy check
- FYEO-THEORIQ-03 – Missing zero checks

Based on our review process, we conclude that the reviewed code implements the documented functionality.

Scope and Rules of Engagement

The FYEO Review Team performed a Security Code Review Theoriq. The following table documents the targets in scope for the engagement. No additional systems or resources were in scope for this assessment.

The source code was supplied through a private repository at https://github.com/chain-ml/theoriq-smart-contracts with the branch: *update-23-07*.

Remediations were done with commit hash 72107f6b0e67d265e1907d85b3413cf52eb9c217.

Files included in the code review

theoriq-smart-contracts/
└── contracts/
  └── contracts/
  ├── AgentToken.sol
  ├── BaseContract.sol
  ├── CreditToken.sol
  ├── Escrow.sol
  ├── FixedPointMath.sol
  ├── ITAIReceiver.sol
  ├── Proxy.sol
  ├── Registry.sol
  └── TheoriqToken.sol

Table 1: Scope

Technical Analyses and Findings

During the Security Code Review Theoriq, we discovered:

- 3 findings with INFORMATIONAL severity rating.

The following chart displays the findings by severity.

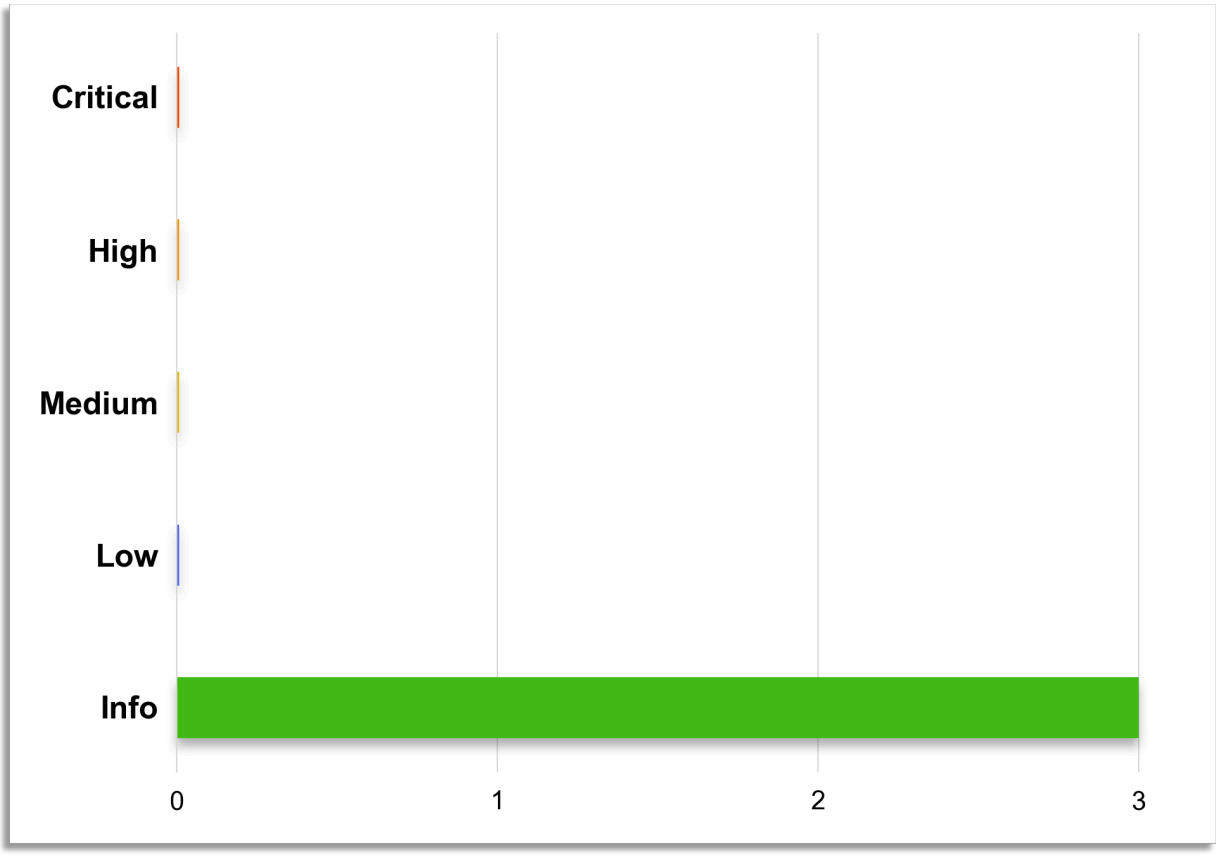

Figure 1: Findings by Severity

Findings

The *Findings* section provides detailed information on each of the findings, including methods of discovery, explanation of severity determination, recommendations, and applicable references.

The following table provides an overview of the findings.

Finding #	Severity	Description
FYEO-THEORIQ-01	**Informational**	Input arrays may be of different lengths
FYEO-THEORIQ-02	**Informational**	Missing reentrancy check
FYEO-THEORIQ-03	**Informational**	Missing zero checks

Table 2: Findings Overview

Technical Analysis

The source code has been manually validated to the extent that the state of the repository allowed. The validation includes confirming that the code correctly implements the intended functionality.

Conclusion

Based on our review process, we conclude that the code implements the documented functionality to the extent of the reviewed code.

Technical Findings

General Observations

The Theoriq program utilizes smart contracts to facilitate transactions where users or AI agents pay other AI agents to perform jobs. The system operates on an EVM-compatible blockchain and includes four primary smart contracts: the registry, the agent and credit tokens and an escrow. All contracts can be updated, paused if needed, and have controlled access based on roles. The Registry contract is responsible for banning users, AgentToken signifies ownership of AI agents, CreditToken helps cover transaction costs, and Escrow handles the financial transactions for agent operations. Technically, any ERC20 token can be used for payments, but the admin maintains a whitelist to approve specific ones.

The updated codebase introduces several improvements to the Theoriq ecosystem. The contracts have been refactored to enhance consistency and readability, making the system more maintainable. New events have been added throughout, improving transparency and making it easier to track important changes in the system. A significant addition is the new Theoriq Token, which brings a native financial layer to the platform. This token features a fixed maximum supply, controlled minting processes, and the ability to efficiently distribute tokens to multiple recipients at once. The token is designed to work seamlessly with the existing components of the Theoriq system, particularly in enforcing platform-wide rules like user banning. Furthermore, the token contract is built with upgradeability in mind, allowing for future enhancements without disrupting the token's functionality or user holdings.

These updates reflect Theoriq's commitment to creating a robust, flexible, and user-friendly ecosystem for AI agent transactions.

Input arrays may be of different lengths

Finding ID: FYEO-THEORIQ-01
Severity: **Informational**
Status: **Remediated**

Description

Two arrays are used in conjunction and are expected to have the same length. This is not checked however.

Proof of Issue

File name: contracts/contracts/Escrow.sol
Line number: 343

```
function onTAITokenReceived(
    address[] memory recipients,
    uint256[] memory amounts
) external nonReentrant returns (bytes4) {
    address receivedTokenAddress = _msgSender();

    if (receivedTokenAddress != address(_taiToken)) {
        revert InvalidTAIContract(receivedTokenAddress);
    }

    for (uint256 i = 0; i < recipients.length; ) {
        _balances[receivedTokenAddress][recipients[i]]
            .totalBalance += amounts[i];
        unchecked {
            ++i;
        }
    }
```

Severity and Impact Summary

Not a security concern.

Recommendation

The implemented caller does check that these arrays are the same length. However, this is a public function and if incorrectly implemented elsewhere, it could trigger errors for out of bounds access.

Missing reentrancy check

Finding ID: FYEO-THEORIQ-02
Severity: **Informational**
Status: **Remediated**

Description

The TheoriqToken's `safeBulkMint` function is missing a reentrancy check. In comparison, the other tokens do implement this check.

Proof of Issue

File name: contracts/contracts/TheoriqToken.sol
Line number: 148

```
function safeBulkMint(
    address receiver,
    address[] memory recipients,
    uint256[] memory amounts
) public onlyRole(MINTER_ROLE) ifNotBanned(receiver) {
```

Severity and Impact Summary

This is an authorized function, but it does call `onTAITokenReceived`.

Recommendation

Add a reentrancy guard.

Missing zero checks

Finding ID: FYEO-THEORIQ-03
Severity: **Informational**
Status: **Remediated**

Description

There are a few instances where addresses should be checked against the zero address.

Proof of Issue

File name: contracts/contracts/TheoriqToken.sol
Line number: 60

```
function initialize(
    ...
    address registry,
    ...
) public initializer {
    ...
    _registry = Registry(registry);
```

The `_registry` can be a zero address.

File name: contracts/contracts/Escrow.sol
Line number: 235

```
function initialize(
    ...
    address taiTokenAddress
) public initializer {
    ...
    _taiToken = ERC20BurnableUpgradeable(taiTokenAddress);
```

The `_taiToken` can be a zero address.

Severity and Impact Summary

By mistake the contracts could be initialized with the wrong values.

Recommendation

Make sure to add checks against the zero address where required.

Our Process

Methodology

FYEO Inc. uses the following high-level methodology when approaching engagements. They are broken up into the following phases.

Figure 2: Methodology Flow

Kickoff

The project is kicked off as the sales process has concluded. We typically set up a kickoff meeting where project stakeholders are gathered to discuss the project as well as the responsibilities of participants. During this meeting we verify the scope of the engagement and discuss the project activities. It's an opportunity for both sides to ask questions and get to know each other. By the end of the kickoff there is an understanding of the following:

- Designated points of contact
- Communication methods and frequency
- Shared documentation
- Code and/or any other artifacts necessary for project success
- Follow-up meeting schedule, such as a technical walkthrough
- Understanding of timeline and duration

Ramp-up

Ramp-up consists of the activities necessary to gain proficiency on the project. This can include the steps needed for familiarity with the codebase or technological innovation utilized. This may include, but is not limited to:

- Reviewing previous work in the area including academic papers
- Reviewing programming language constructs for specific languages
- Researching common flaws and recent technological advancements

Review

The review phase is where most of the work on the engagement is completed. This is the phase where we analyze the project for flaws and issues that impact the security posture. Depending on the project this may include an analysis of the architecture, a review of the code, and a specification matching to match the architecture to the implemented code.

In this code audit, we performed the following tasks:

1. Security analysis and architecture review of the original protocol
2. Review of the code written for the project
3. Compliance of the code with the provided technical documentation

The review for this project was performed using manual methods and utilizing the experience of the reviewer. No dynamic testing was performed, only the use of custom-built scripts and tools were used to assist the reviewer during the testing. We discuss our methodology in more detail in the following sections.

Code Safety

We analyzed the provided code, checking for issues related to the following categories:

- General code safety and susceptibility to known issues
- Poor coding practices and unsafe behavior
- Leakage of secrets or other sensitive data through memory mismanagement
- Susceptibility to misuse and system errors
- Error management and logging

This list is general and not comprehensive, meant only to give an understanding of the issues we are looking for.

Technical Specification Matching

We analyzed the provided documentation and checked that the code matches the specification. We checked for things such as:

- Proper implementation of the documented protocol phases
- Proper error handling
- Adherence to the protocol logical description

Reporting

FYEO Inc. delivers a draft report that contains an executive summary, technical details, and observations about the project.

The executive summary contains an overview of the engagement including the number of findings as well as a statement about our general risk assessment of the project. We may conclude that the overall risk is low but depending on what was assessed we may conclude that more scrutiny of the project is needed.

We report security issues identified, as well as informational findings for improvement, categorized by the following labels:

- Critical
- High
- Medium
- Low
- Informational

The technical details are aimed more at developers, describing the issues, the severity ranking and recommendations for mitigation.

As we perform the audit, we may identify issues that aren't security related, but are general best practices and steps that can be taken to lower the attack surface of the project. We will call those out as we encounter them and as time permits.

As an optional step, we can agree on the creation of a public report that can be shared and distributed with a larger audience.

Verify

After the preliminary findings have been delivered, this could be in the form of the approved communication channel or delivery of the draft report, we will verify any fixes within a window of time specified in the project. After the fixes have been verified, we will change the status of the finding in the report from open to remediated.

The output of this phase will be a final report with any mitigated findings noted.

Additional Note

It is important to note that, although we did our best in our analysis, no code audit or assessment is a guarantee of the absence of flaws. Our effort was constrained by resource and time limits along with the scope of the agreement.

While assessing the severity of the findings, we considered the impact, ease of exploitability, and the probability of attack. This is a solid baseline for severity determination.

The Classification of vulnerabilities

Security vulnerabilities and areas for improvement are weighted into one of several categories using, but is not limited to, the criteria listed below:

Critical – vulnerability will lead to a loss of protected assets

- This is a vulnerability that would lead to immediate loss of protected assets
- The complexity to exploit is low
- The probability of exploit is high

High - vulnerability has potential to lead to a loss of protected assets

- All discrepancies found where there is a security claim made in the documentation that cannot be found in the code
- All mismatches from the stated and actual functionality
- Unprotected key material
- Weak encryption of keys
- Badly generated key materials
- Txn signatures not verified
- Spending of funds through logic errors
- Calculation errors overflows and underflows

Medium - vulnerability hampers the uptime of the system or can lead to other problems

- Insecure calls to third party libraries
- Use of untested or nonstandard or non-peer-reviewed crypto functions
- Program crashes, leaves core dumps or writes sensitive data to log files

Low – vulnerability has a security impact but does not directly affect the protected assets

- Overly complex functions
- Unchecked return values from 3rd party libraries that could alter the execution flow

Informational

- General recommendations