Audit Report

tokenfactory Changes and fiattokenfactory

v1.0

January 17, 2024

Table of Contents

License

Disclaimer

THE CONTENT OF THIS AUDIT REPORT IS PROVIDED “AS IS”, WITHOUT REPRESENTATIONS AND WARRANTIES OF ANY KIND.

THE AUTHOR AND HIS EMPLOYER DISCLAIM ANY LIABILITY FOR DAMAGE ARISING OUT OF, OR IN CONNECTION WITH, THIS AUDIT REPORT.

THIS AUDIT REPORT IS ADDRESSED EXCLUSIVELY TO THE CLIENT. THE AUTHOR AND HIS EMPLOYER UNDERTAKE NO LIABILITY OR RESPONSIBILITY TOWARDS THE CLIENT OR THIRD PARTIES.

This audit has been performed by

Oak Security

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of This Report

Oak Security has been engaged by NASD Inc to perform a security audit of changes to the `tokenfactory` module and the `fiattokenfactory` module of the Noble chain.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the following target:

Repository	https://github.com/strangelove-ventures/noble/
Commit	`c3fa39277956413e334b0050a08bb848b85464f1`
Scope	The scope of this audit was limited to the changes to the `tokenfactory` module in the `x/tokenfactory` directory since our last audit, which has been performed at commit `eab846178aef764c925c7f1211e93af6b5828b72`, as well as the `fiattokenfactory` module in the `x/fiattokenfactory` directory.

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line-by-line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
4. Report preparation

Functionality Overview

Noble is an application-specific Cosmos SDK blockchain purpose-built for native asset issuance. The `tokenfactory` and `fiattokenfactory` modules enable the minting of generic assets, controlled by the asset issuers, and separate and distinct from the governance of the Noble chain.

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending,** **Acknowledged**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than in a security audit and vice versa.

Code Quality Criteria

The auditor team assesses the codebase's code quality criteria as follows:

Criteria	Status	Comment
Code complexity	Low-Medium	-
Code readability and clarity	Medium-High	-
Level of documentation	Medium-High	-
Test coverage	-	-

Summary of Findings

No	Description	Severity	Status
1	Tight coupling of the `Mint` and `Burn` `MsgServer` and `Keeper` functions	Informational	Acknowledged
2	Misleading `simulation` package import identifier for the `fiattokenfactory` module	Informational	Acknowledged

Detailed Findings

1. Tight coupling of the `Mint` and `Burn` `MsgServer` and `Keeper` functions

Severity: Informational

The `Mint` and `Burn` functions in `keeper/msg_server_mint.go` and `keeper/msg_server_burn.go` in both the `tokenfactory` and `fiattokenfactory` modules were refactored by extracting the logic from the `MsgServer` functions to separate `Keeper` functions to expose them to other Cosmos SDK modules within the same chain.

The original `MsgServer` function signatures have been reused. Specifically, the `MsgMint` and `MsgBurn` message types are used as parameters. Moreover, the keeper functions return the same response types as the `MsgServer` functions.

However, it is considered best practice to keep the `MsgServer` and `Keeper` functions decoupled by using explicit function parameters and appropriate return types. This allows for more flexibility in the future, for example, if the `MsgMint` message or `MsgMintResponse` return type needs to be refactored.

Recommendation

We recommend using explicit function parameters for the `Mint` and `Burn` keeper functions and adapting the return types accordingly. For example, the `Burn` keeper function could be refactored to the following:

```
func (k Keeper) Burn(ctx sdk.Context, from string, amount sdk.Coin) error
```

Status: Acknowledged

2. Misleading `simulation` package import identifier for the `fiattokenfactory` module

Severity: Informational

In `x/fiattokenfactory/module_simulation.go:15`, the `simulation` package is imported from the `fiattokenfactory` module with the `tokenfactorysimulation` identifier. However, this identifier is misleading, as the `simulation` package is not part of the `tokenfactory` module but rather the `fiattokenfactory` module.

Recommendation

We recommend renaming the import identifier to `fiattokenfactorysimulation` to avoid confusion.

Status: Acknowledged