

September 26, 2024

# Scribe

## Smart Contract Security Assessment



## Contents

<b>About Zellic</b>	<b>3</b>
<hr/>	
<b>1. Overview</b>	<b>3</b>
1.1. Executive Summary	4
1.2. Goals of the Assessment	4
1.3. Non-goals and Limitations	4
1.4. Results	4
<hr/>	
<b>2. Introduction</b>	<b>5</b>
2.1. About Scribe	6
2.2. Methodology	6
2.3. Scope	8
2.4. Project Overview	8
2.5. Project Timeline	9
<hr/>	
<b>3. Changes</b>	<b>9</b>
3.1. Renaming variables, events, contracts, and comments	10
3.2. Removing register, assign, and withdraw functions	10
3.3. Removing some events	10
3.4. Removing the PositionHelper contract	10
<hr/>	
<b>4. Assessment Results</b>	<b>10</b>
4.1. Disclaimer	11

## About Zellic

Zellic is a vulnerability research firm with deep expertise in blockchain security. We specialize in EVM, Move (Aptos and Sui), and Solana as well as Cairo, NEAR, and Cosmos. We review L1s and L2s, cross-chain protocols, wallets and applied cryptography, zero-knowledge circuits, web applications, and more.

Prior to Zellic, we founded the [#1 CTF \(competitive hacking\) team](#) worldwide in 2020, 2021, and 2023. Our engineers bring a rich set of skills and backgrounds, including cryptography, web security, mobile security, low-level exploitation, and finance. Our background in traditional information security and competitive hacking has enabled us to consistently discover hidden vulnerabilities and develop novel security research, earning us the reputation as the go-to security firm for teams whose rate of innovation outpaces the existing security landscape.

For more on Zellic's ongoing security research initiatives, check out our website [zellic.io](https://zellic.io) and follow [@zellic\\_io](#) on Twitter. If you are interested in partnering with Zellic, contact us at [hello@zellic.io](mailto:hello@zellic.io).



## 1. Overview

### 1.1. Executive Summary

Zellic conducted a security assessment for Scribe.Exchange from September 19th to September 20th, 2024. During this engagement, Zellic reviewed Scribe's code for security vulnerabilities, design issues, and general weaknesses in security posture.

---

### 1.2. Goals of the Assessment

For this audit, we compared the scope of this audit to the specified commit hash ([aa294d72 7](#)) of Kim Protocol. In section [3. 7](#), we offer an overview of the changes.

In a security assessment, goals are framed in terms of questions that we wish to answer. These questions are agreed upon through close communication between Zellic and the client. In this assessment, we sought to answer the following questions:

- Do the changes cause any logical problems?
  - Do the changes cause any potential security vulnerabilities?
  - Do the changes break the functionality of the contract?
- 

### 1.3. Non-goals and Limitations

We did not assess the following areas that were outside the scope of this engagement:

- Front-end components
- Infrastructure relating to the project
- Key custody
- Issues not caused by the changes

Due to the time-boxed nature of security assessments in general, there are limitations in the coverage an assessment can provide.

---

### 1.4. Results

During our assessment on the scoped Scribe contracts, there were no security vulnerabilities discovered.

Breakdown of Finding Impacts

Impact Level	Count
<div><div></div>Critical</div>	0
<div><div></div>High</div>	0
<div><div></div>Medium</div>	0
<div><div></div>Low</div>	0
<div><div></div>Informational</div>	0

## 2. Introduction

### 2.1. About Scribe

Scribe.Exchange contributed the following description of Scribe:

Scribe Protocol is an AMM DEX with concentrated liquidity pools with hooks known as plugins.

---

### 2.2. Methodology

During a security assessment, Zellic works through standard phases of security auditing, including both automated testing and manual review. These processes can vary significantly per engagement, but the majority of the time is spent on a thorough manual review of the entire scope.

Alongside a variety of tools and analyzers used on an as-needed basis, Zellic focuses primarily on the following classes of security and reliability issues:

**Basic coding mistakes.** Many critical vulnerabilities in the past have been caused by simple, surface-level mistakes that could have easily been caught ahead of time by code review. Depending on the engagement, we may also employ sophisticated analyzers such as model checkers, theorem provers, fuzzers, and so on as necessary. We also perform a cursory review of the code to familiarize ourselves with the contracts.

**Business logic errors.** Business logic is the heart of any smart contract application. We examine the specifications and designs for inconsistencies, flaws, and weaknesses that create opportunities for abuse. For example, these include problems like unrealistic tokenomics or dangerous arbitrage opportunities. To the best of our abilities, time permitting, we also review the contract logic to ensure that the code implements the expected functionality as specified in the platform's design documents.

**Integration risks.** Several well-known exploits have not been the result of any bug within the contract itself; rather, they are an unintended consequence of the contract's interaction with the broader DeFi ecosystem. Time permitting, we review external interactions and summarize the associated risks: for example, flash loan attacks, oracle price manipulation, MEV/sandwich attacks, and so on.

**Code maturity.** We look for potential improvements in the codebase in general. We look for violations of industry best practices and guidelines and code quality standards. We also provide suggestions for possible optimizations, such as gas optimization, upgradability weaknesses, centralization risks, and so on.

For each finding, Zellic assigns it an impact rating based on its severity and likelihood. There is no hard-and-fast formula for calculating a finding's impact. Instead, we assign it on a case-by-case basis based on our judgment and experience. Both the severity and likelihood of an issue affect its impact. For instance, a highly severe issue's impact may be attenuated by a low likelihood. We assign the following impact ratings (ordered by importance): Critical, High, Medium, Low, and

### Informational.

Zellic organizes its reports such that the most important findings come first in the document, rather than being strictly ordered on impact alone. Thus, we may sometimes emphasize an "Informational" finding higher than a "Low" finding. The key distinction is that although certain findings may have the same impact rating, their *importance* may differ. This varies based on various soft factors, like our clients' threat models, their business needs, and so on. We aim to provide useful and actionable advice to our partners considering their long-term goals, rather than a simple list of security issues at present.

## 2.3. Scope

The engagement involved a review of the following targets:

### Scribe Contracts

Type	Solidity
Platform	EVM-compatible
Target	ScribePoolFactory
Repository	<a href="https://github.com/scribe-dex/ScribePoolFactory">https://github.com/scribe-dex/ScribePoolFactory</a> ↗
Version	62dd301e1065246e2482c97edd1ed079a25130e1
Programs	Dividends.sol NFTPool.sol NFTPoolFactory.sol NitroPool.sol NitroPoolFactory.sol ProtocolEarnings.sol ScribeMaster.sol YieldBooster.sol

## 2.4. Project Overview

Zellic was contracted to perform a security assessment for a total of 1.5 person-days. The assessment was conducted by two consultants over the course of two calendar days.

### Contact Information



---

The following project managers were associated with the engagement:

**Jacob Goreski**  
↗ Engagement Manager  
[jacob@zellic.io](mailto:jacob@zellic.io) ↗

**Chad McDonald**  
↗ Engagement Manager  
[chad@zellic.io](mailto:chad@zellic.io) ↗

---

The following consultants were engaged to conduct the assessment:

**Jinseo Kim**  
↗ Engineer  
[jinseo@zellic.io](mailto:jinseo@zellic.io) ↗

**Juchang Lee**  
↗ Engineer  
[lee@zellic.io](mailto:lee@zellic.io) ↗

---

## 2.5. Project Timeline

The key dates of the engagement are detailed below.

---

<b>September 19, 2024</b>	Start of primary review period
---------------------------	--------------------------------

---

<b>September 20, 2024</b>	End of primary review period
---------------------------	------------------------------

## 3. Changes

This section documents the exact changes of the codebase that were considered in scope for this audit.

---

### 3.1. Renaming variables, events, contracts, and comments

Most of the changes are renaming. Across the board, the name "Kim" in variables, events, contracts, and comments has been replaced with "Scribe". This change does not affect the logic.

---

### 3.2. Removing register, assign, and withdraw functions

The functions `register`, `assign`, and `withdraw` are removed. These functions implement the interface for the Sequencer Fee Sharing feature of Mode blockchain, which Kim Protocol resided in. This change affects the contracts below:

- Dividends
  - NFTPool
  - NFTPoolFactory
  - NitroPool
  - NitroPoolFactory
  - ProtocolEarnings
  - ScribeMaster
  - YieldBooster
- 

### 3.3. Removing some events

The events `SetDefaultFee`, `SetFeeAddress`, `SetEmergencyRecoveryAddress`, `SetExemptedAddress`, and `SetNitroPoolOwner` are removed from the contract `NitroPoolFactory`, without involving any logic change.

---

### 3.4. Removing the PositionHelper contract

The `PositionHelper` contract was removed. Since it was an auxiliary contract that provides another interface for the existing feature, the overall logic of the protocol is not affected.

## 4. Assessment Results

At the time of our assessment, the reviewed code was not deployed to the Ethereum Mainnet.

During our assessment on the scoped Scribe contracts, there were no security vulnerabilities discovered.

---

### 4.1. Disclaimer

This assessment does not provide any warranties about finding all possible issues within its scope; in other words, the evaluation results do not guarantee the absence of any subsequent issues. Zellic, of course, also cannot make guarantees about any code added to the project after the version reviewed during our assessment. Furthermore, because a single assessment can never be considered comprehensive, we always recommend multiple independent assessments paired with a bug bounty program.

For each finding, Zellic provides a recommended solution. All code samples in these recommendations are intended to convey how an issue may be resolved (i.e., the idea), but they may not be tested or functional code. These recommendations are not exhaustive, and we encourage our partners to consider them as a starting point for further discussion. We are happy to provide additional guidance and advice as needed.

Finally, the contents of this assessment report are for informational purposes only; do not construe any information in this report as legal, tax, investment, or financial advice. Nothing contained in this report constitutes a solicitation or endorsement of a project by Zellic.