Zellic

Prepared for
Mohadang
Scribe.Exchange

Prepared by
Nipun Gupta
Sunwoo Hwang
Zellic

October 10, 2024

Scribe

Smart Contract Security Assessment

Contents

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 ↗ and follow @zellic_io ↗ on Twitter. If you are interested in partnering with Zellic, contact us at hello@zellic.io ↗.

1. Overview

1.1. Executive Summary

Zellic conducted a security assessment for Scribe.Exchange from October 4th to October 7th, 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 its scope to the contracts KimToken and xKimToken deployed at addresses `0x6863fb62Ed27A9DdF458105B507C15b5d741d62e` and `0x4D850FE01F07BEd416558A34dbde88bA60aE19BE` on the Mode network, respectively. An overview of the changes can be found in section 4. ↗.

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 introduce any logical issues?
- Do the changes create any new potential security vulnerabilities?
- Do the changes compromise 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.

Zellic recorded its notes and observations from the assessment for the benefit of Scribe.Exchange

in the Discussion section (3. ↗).

Breakdown of Finding Impacts

Impact Level	Count
Critical	0
High	0
Medium	0
Low	0
Informational	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.

Finally, Zellic provides a list of miscellaneous observations that do not have security impact or are not directly related to the scoped contracts itself. These observations — found in the Discussion (3. ↗) section of the document — may include suggestions for improving the codebase, or general recommendations, but do not necessarily convey that we suggest a code change.

2.3. Scope

The engagement involved a review of the following targets:

Scribe Contracts

Type	Solidity
Platform	EVM-compatible
Target	scribe-tokens
Repository	https://github.com/scribe-dex/scribe-tokens ↗
Version	`33abd392eb57e3df110551babc1bc36339da14c3`
Programs	`ScribeToken` `XScribeToken`

2.4. Project Overview

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

Contact Information

The following project manager was associated with the engagement:

Chad McDonald
Engagement Manager
chad@zellic.io

The following consultants were engaged to conduct the assessment:

Nipun Gupta
Engineer
nipun@zellic.io

Sunwoo Hwang
Engineer
sunwoo@zellic.io

2.5. Project Timeline

The key dates of the engagement are detailed below.

October 4, 2024	Kick-off call
October 4, 2024	Start of primary review period
October 7, 2024	End of primary review period

3. Discussion

The purpose of this section is to document miscellaneous observations that we made during the assessment. These discussion notes are not necessarily security related and do not convey that we are suggesting a code change.

3.1. Redeeming process breaks if `address(0)` is not a `_transferWhitelist` address

During the finalization of the redeeming process, the function `_finalizeRedeem` calls `_burn`, which internally calls `_beforeTokenTransfer` with the `to` address as `address(0)`. This implies that `address(0)` should always remain in the `_transferWhitelist`. Therefore, calls to `_finalizeRedeem` will fail before the owner adds `address(0)` to the `_transferWhitelist`. As `_beforeTokenTransfer` allows the `from` address to be `address(0)` for minting, as shown below, it would be convenient to allow `address(0)` as the `to` address for the burning of tokens.

```
function _beforeTokenTransfer(
    address from,
    address to,
    uint256 /*amount*/
) internal view override {
    require(
        from == address(0) ||
            _transferWhitelist.contains(from) ||
            _transferWhitelist.contains(to),
        "transfer: not allowed"
    );
}
```

Remediation

This issue has been acknowledged by Scribe.Exchange.

Additionally, the Scribe team has stated that they'll be keeping `address(0)` in the whitelist.

3.2. Inaccurate comments

The function `withdraw` in XScribeToken and ScribeToken contracts calls `feeSharingContract` using the function selector `0xe63697c8`. Although the function selector is accurate, the comments above the call imply that the function selector is `bytes4(keccak256(bytes('withdraw(uint256, address, uint256)')));`, which is inaccurate. While calculating the function selector, the spaces

between the argument types are removed, and thus the actual function selection is derived as follows:

```
bytes4(keccak256(bytes('withdraw(uint256,address,uint256)')));
```

Remediation

This issue has been acknowledged by Scribe.Exchange, and a fix was implemented in commit 6fe8a2b2 ↗.

4. Changes

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

4.1. Renaming variables, events, contracts, and comments

All 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 of the contracts.

5. 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.

5.1. 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.