Zellic

TrustToken

Smart Contract Security Assessment

October 25, 2022

Prepared for:

Christian Alexander Boehnke de Lorraine-Elbeuf

TrueFi

Prepared by:

Katerina Belotskaia

Zellic Inc.

Contents

About Zellic 2

1 Executive Summary 3

2 Introduction 4

2.1 About TrustToken 4

2.2 Methodology 4

2.3 Scope 5

2.4 Project Overview 5

2.5 Project Timeline 6

3 Detailed Findings 7

3.1 `transfer` function usage 7

4 Audit Results 8

4.1 Disclaimers 8

About Zellic

Zellic was founded in 2020 by a team of blockchain specialists with more than a decade of combined industry experience. We are leading experts in smart contracts and Web3 development, cryptography, web security, and reverse engineering. Before Zellic, we founded perfect blue, the top competitive hacking team in the world. Since then, our team has won countless cybersecurity contests and blockchain security events.

Zellic aims to treat clients on a case-by-case basis and to consider their individual, unique concerns and business needs. Our goal is to see the long-term success of our partners rather than simply provide a list of present security issues. Similarly, we strive to adapt to our partners' timelines and to be as available as possible. To keep up with our latest endeavors and research, check out our website zellic.io or follow @zellic_io on Twitter. If you are interested in partnering with Zellic, please email us at hello@zellic.io or contact us on Telegram at https://t.me/zellic_io.

1 Executive Summary

Zellic conducted an audit for TrueFi from August 29th to August 30st, 2022.

Our general overview of the code is that it was very well-organized and structured. The code was easy to comprehend, and in most cases, intuitive.

We applaud TrueFi for their attention to detail and diligence in maintaining incredibly high code quality standards in the development of TrustToken.

Zellic thoroughly reviewed the TrustToken codebase to find protocol-breaking bugs as defined by the documentation and to find any technical issues outlined in the Methodology section (2.2) of this document.

Specifically, taking into account TrustToken's threat model, we focused heavily on issues that would break core invariants such as bypassing the protection that was put in place for the mint, transfer and burn functions of the TrustToken.

During our assessment on the scoped TrustToken contracts, we discovered one findings. Fortunately, no critical issues were found. This finding was informational in nature.

Breakdown of Finding Impacts

Impact Level	Count
Critical	0
High	0
Medium	0
Low	0
Informational	1

2 Introduction

2.1 About TrustToken

TrustToken is a DeFi protocol that provides uncollateralized loans to institutional investors using an on-chain credit score mechanism. Users can deposit stablecoins to TrueFi lending pools and earn yields.

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 open-source 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. We analyze the scoped smart contract code using automated tools to quickly sieve out and catch these shallow bugs. Depending on the engagement, we may also employ sophisticated analyzers such as model checkers, theorem provers, fuzzers, and so forth 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 manually review the contract logic to ensure that the code implements the expected functionality as specified in the platform's design documents. We also thoroughly examine the specifications and designs themselves for inconsistencies, flaws, and vulnerabilities. This involves use cases that open the opportunity for abuse, such as flawed tokenomics or share pricing, arbitrage opportunities, and so forth.

Complex integration risks. Several high-profile 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. We perform a meticulous review of all of the contract's possible external interactions and summarize the associated risks: for example, flash loan attacks, oracle price manipulation, MEV/sandwich attacks, and so forth.

Code maturity. We review for possible 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, upgradeability weaknesses, centralization risks, and so forth.

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; we assign it on a case-by-case basis based on our professional judgment and experience. As one would expect, both the severity and likelihood of an issue affect its impact; for instance, a highly severe issue's impact may be attenuated by a very low likelihood. We assign the following impact ratings (ordered by importance): Critical, High, Medium, Low, and Informational.

Similarly, Zellic organizes its reports such that the most important findings come first in the document rather than being 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 numerous soft factors, such as our clients' threat models, their business needs, their project timelines, and so forth. We aim to provide useful and actionable advice to our partners that consider their long-term goals rather than simply provide a list of security issues at present.

2.3 Scope

The engagement involved a review of the following targets:

TrustToken Contracts

Repository	https://github.com/trusttoken/contracts-tequila
Versions	`6a9b3bdaf9c58fcf692657e8af9e0fc9bb5b2f84`
Programs	• ./DaoTreasury.sol • ./helium/access/Upgradeable.sol • ./helium/proxy/ProxyWrapper.sol
Type	Solidity
Platform	EVM-compatible

2.4 Project Overview

Zellic was contracted to perform a security assessment with one consultant for a total of one person-day. The assessment was conducted over the course of one day.

Contact Information

The following project managers were associated with the engagement:

Jasraj Bedi, Co-founder
jazzy@zellic.io

Stephen Tong, Co-founder
stephen@zellic.io

The following consultants were engaged to conduct the assessment:

Katerina Belotskaia, Engineer
kate@zellic.io

2.5 Project Timeline

The key dates of the engagement are detailed below.

August 29, 2022	Start of primary review period
August 30, 2022	End of primary review period

3 Detailed Findings

3.1 `transfer` function usage

- **Target**: DaoTreasury
- **Category**: Business Logic
- **Likelihood**: N/A
- **Severity**: Informational
- **Impact**: Informational

Description

The `transferEth` function calls the `transfer` function to send requested Ether amount.

```
function transferEth(address payable to, uint256 amount) public onlyRole(
    MANAGER_ROLE) whenNotPaused {
        to.transfer(amount);
        ...
    }
```

Impact

The `transfer` function uses a hardcoded amount of GAS and will fail if GAS costs increase in the future, so it is no longer recommended for use.

Recommendations

Consider using `to.call{value: amount}("")` function:

```
function transferEth(address payable to, uint256 amount) public onlyRole(
    MANAGER_ROLE) whenNotPaused {
    (bool sent, bytes memory data) = to.call{value: amount}("");
    require(sent, "Failed to send Ether");
    ...
```

Remediation

The issue has been acknowledged by TrueFi. The TrueFi decided not to support ETH transfers and functionality was deleted in commit 0b0ca5f.

4 Audit Results

At the time of our audit, the code was not deployed to mainnet evm.

During our audit, we discovered one finding. This one finding was suggestions (informational). TrueFi acknowledged all findings and implemented fixes.

4.1 Disclaimers

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 additional code added to the assessed project after the audit version of 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 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.

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.