Security assessment and code review

Initial Delivery: July 12, 2021
Updates: July 22-27, July 30 2021
Camera Ready: September 8, 2021

Prepared for:
Hsuan-Ting Chu | Dinngo Dev

Prepared by:
Mikerah Quintyne-Collins | HashCloak Inc

Table Of Contents

Executive Summary

Trevi is a staking-based rewards program offered as part of Dinngo Dev's Furucombo offering. Trevi consists of three actors in its system: Fountains, Angels and Archangels. Fountains are vaults in which users store their tokens. Angels are reward managers that decide how staking rewards can be disbursed to users. Angels can be deployed by anyone. Finally, we have Archangels that manage the relationships between Fountains and Angels.

From July 5, 2021 to July 9, 2021, The Dinngo Dev team engaged HashCloak for an audit of Trevi, their ERC20-based staking protocol. The audit was done with 1 auditor over 1 person week. The Trevi codebase was assessed at commit b3f7fd332873321152db48c9d43fc23a60a29f1a. After the initial report was delivered, the Dinngo Dev team updated the codebase with our suggestions. The new commit is 2a94aa1ba2e9a1a5cd9d5d9bd5256f3fedc2ab2f with additional changes made to the README with no modifications to the codebase at commit d31131dd2ac61f509b57813c29c4828ae27c1b12. These fixes were revised from July 19, 2021 to July 22, 2021 with 1 auditor over 1 person week. While preparing the publication of this document, the developers brought up an issue and revised a fix at commit 6d68386b4077a1e29386344369ab9b4bc7f8af89. The fix was revised with 1 auditor over 1 day. While preparing the deployment of the Trevi contracts in September, the development team reached out to HashCloak for final checks and comments. We reviewed commits ff4b73c514c7ffa7dea849d11febc6ae643c5a29 and 5fd9dbc55fc4fa91141f2c4468b85728a3f77c9e on September 7 and 8 with 1 auditor over 1 day.

The scope were all files ending in **.sol** in the main repository. It is assumed that libraries and interfaces were implemented correctly and thus outside the scope of the audit.

During the first part of the audit, we familiarized ourselves with the Trevi smart contracts. In the second part, we manually reviewed and used off-the-shelf automated analysis tools for the audit. During the second short assessment, during the period of July 19, 2021 to July 22, 2021, the development team found an issue relating to the use of deflationary tokens. We revised their fix and found no extra issues. During the most recent assessment, we found no extra issues affecting the security of the Trevi codebase although we gave a suggestion to add more details upon the failure of a method call in an emergency withdrawal situation.

We found a variety of issues ranging from medium to informational.

Severity	Number of Findings
Critical	0
High	1
Medium	3
Low	1
Informational	2

Overview

The Trevi protocol is made up of the following smart contracts

- Fountain.sol A vault which is setup per token, stores users' funds and issues them FTN tokens for staking
- FountainFactory.sol Fountains are deployed with this factory contract
- FountainToken.sol An ERC20 implementation of the FTN staking token with permits
- FountainBase.sol An abstract Fountain implementation
- Archangel.sol Manages the relationship between Fountains and Angels and flash loan related functionality
- Angel.sol A fork of MiniChefV2 from SushiSwap. It provides rewards functionality that is controlled by Fountains.
- AngelFactory.sol Angels are deployed with this factory contract
- AngelBase.sol An abstract Angel implementation
- JoinPermit.sol A Fountain in which a sender can join for a user before a set deadline
- HarvestPermit.sol A Fountain in which a sender can harvest for a user before a set deadline
- ERC20FlashLoan.sol An ERC3156-compatible contracts for ERC20-based flash loans
- ERC20.sol A fork of OpenZeppelin's ERC20 implementation with changes to the _burn internal method
- ERC20Permit.sol Inherits from Trevi's ERC20.sol fork and adds support for EIP 2612

Findings

Fountains and Angels don’t consider deflationary tokens

Type: High Severity
Files affected: FountainBase.sol, AngelBase.sol

Since the Fountain and Angel don’t explicitly check for certain behaviors of an ERC20 token, this allows for a deflationary token to be added to the Fountain or Angel. Deflationary tokens are ERC20 in which there are fees upon transfer of the tokens. As such, this can be used as a way to potentially drain funds handled by Fountains.

Impact: Funds drained from Fountains.

Suggestion: Add a whitelist of known ERC20 compliant tokens to both the smart contracts and Trevi UI.

Status: The developers will be updating the Trevi UI to caution users about the use of deflationary tokens. Due to the permissionless nature of Trevi, the developers have decided to forego an on-chain whitelist of ERC20-compliant tokens in order to mitigate this issue. Instead, they carefully calculated the exact amount of tokens deposited/withdrawn into/out of the Fountains. This fix was applied at commit 6d68386b4077a1e29386344369ab9b4bc7f8af89

Check for duplicate LP tokens in AngelBase.sol

Type: Medium Severity
Files affected: AngelBase.sol

When adding a new LP token to an Angel, a malicious owner can duplicate i.e. re-add this LP token to the Angel. This leads to rewards being miscalculated.

Impact: Duplicating LP tokens for an Angel will lead to miscalculations of rewards for users.

Suggestion: Check for duplicate LP tokens upon calling `add()`. If the LP token is a duplicate, revert with an error. Otherwise, proceed as usual.

Status: After discussions with the development team, upon calling `setPoolId()` in FountainBase.sol, the function should revert if the same Angel has previously called this function with the same pool id. As such, the code stays the same.

Potential for re-entrancy bugs in FountainBase.sol

Type: Medium Severity
Files affected: FountainBase.sol

In FountainBase.sol, there are several functions handling token transfers for which re-entrancy is possible. The following functions are susceptible to this:

- FountainBase._depositAngel()
- FountainBase._emergencyWithdrawAngel()
- FountainBase.deposit()
- FountainBase.depositTo()
- FountainBase.emergencyWithdraw()
- FountainBase.harvest()
- FountainBase.withdraw()
- FountainBase.withdrawTo()

Impact: Might lead to unexpected behaviors that result in the loss of funds.

Suggestion: Use a re-entrancy guard for FountainBase.sol

Status: After discussions with the development team, it was determined that this was a false positive as the internal functions have the `nonReentrant` modifier applied. However, in order to save gas and to better ensure that these methods are safe from re-entrancy attacks, the development team has applied the `nonReentrant` modifier to the associated external functions instead. These changes were added at commit 9f0c4fe5a6bba71f409f97a1bb0fefce0eda6cd5.

Potential for re-entrancy bugs in HarvestPermit.sol

Type: Medium Severity
Files affected: HarvestPermit.sol

In HarvestPermit.sol, harvestFrom() handles token transfers which make external calls. This may lead to a re-entrancy vulnerability.

Impact: Might lead to unexpected behaviors that result in the loss of funds.
Suggestion: Use a re-entrancy guard for FountainBase.sol of which HarvestPermit.sol is inheriting from

Status: After discussions with the team, this was a false positive from the initial audit report. However, the nonReentrant modifier was applied to the `canHarvestFrom` and `harvestAllFrom` methods for the same reasons as the changes in FountainBase.sol. The commit in which these changes were added is 9f0c4fe5a6bba71f409f97a1bb0fefce0eda6cd5.

Potential for re-entrancy bugs in AngelBase.sol that lead to out of order events

Type: Low Severity
Files affected: AngelBase.sol

In AngelBase.sol, there are several functions for which re-entrancy leading to out of order events is possible. The following functions are susceptible to this:

- AngelBase.deposit()
- AngelBase.emergencyWithdraw()
- AngelBase.harvest()
- AngelBase.withdraw()

Impact: Might lead to unexpected behaviors.
Suggestion: Use a re-entrancy guard for AngelBase.sol

Status: After discussions with the team, it was determined that this vulnerability has minimal effects on their end user. As such, no fixes were applied.

Use rounding libraries to minimize risks of miscalculations due to rounding

Type: Informational
Files affected: FountainBase.sol, AngelBase.sol, Archangel.sol

Throughout the codebase, it is assumed that all calculations involved are whole integers. However, due to the nature of these calculations, this assumption doesn't always hold. As such, there is a potential for rounding errors due to a loss of precision. This can potentially result in users getting more/less tokens than they want.

Impact: May lead to miscalculations
Suggestion: Use rounding libraries to minimize risks of miscalculations due to loss of precision
Status: After discussions with the team, it was determined that the current way these kinds of calculations are done poses little risk. They have since provided more details through comments on ambiguous calculations.

Change SushiSwap related terminology in AngelBase.sol

Type: Informational
Files affected: AngelBase.sol

Since AngelBase.sol is a fork of MiniChefV2, the contract has a lot of SUSHI specific terminology that contain concepts that are not relevant for Trevi.

Impact: Might lead to confusion while reading this contract
Suggestion: Update SUSHI related terminology to fit Trevi's terminology.
Status: All SushiSwap related terminology has been removed.