Guild Gated Contracts Security Audit

Audit Resources:

Github Repository of the project was provided. (Link)

Project Author:

- Guild.xyz (Github)

Project Auditor:

- Umair Mirza (Github)

Table of Contents

Audit Summary

The YieldFarm project has been compiled, deployed and tested using the **Hardhat** smart contract development tool chain. The project is comprised of three smart contracts, namely:

- GatedDistributor.sol
- gatedERC721.sol
- RequestGuildRole.sol

Following libraries and interfaces have been integrated with the smart contracts:

- OpenZeppelin
- Chainlink

The contracts have been audited by 1 resident from October 18th to October 21st. The repository was under active development during the audit.

Scope

The scope of this audit is limited to the smart contracts mentioned above. Frontend modules of the project have not been audited.
The commit that has been audited is: **3df46b420bd6f8806792aee9afe8f380d864fa63**
This audit is about identifying potential vulnerabilities in the smart contracts. The audit may not identify all potential attack vectors or areas of vulnerability.

Code Evaluation Matrix

Category	Mark	Description
Access Control	Good	Access control is appropriate for this contract
Compiler	Medium	Solidity 0.8.17 is used which is quite recent and should not be trusted to deploy on mainnet at the moment
Complexity	Medium	Similar variable names(shadowing) create a bit of confusion while reviewing. This way of writing code can be prone to mistakes later on
Libraries	Good	Openzeppelin and Chainlink have been used for this project which are considered mature and stable libraries.
Decentralisation	Good	The contract follows the rules of decentralisation
Code Stability	Average	The last commit to the repository was on 3rd October 2022 which suggests that the repository is under active development.
Documentation	Good	The project has used solidity-docgen package to generate the documentation which is quite neat. However, more description can be added to the function details.

Monitoring	Good	Events have been added to all important functions in the contract

Findings Description

Findings have been broken down into sections by their respective impact:

- Critical, High, Medium, Low Impact
    - These are findings that range from attacks that may cause loss of funds, impact control/ownership of the contracts, or cause any unintended consequences/actions that are outside the scope of the requirements.
- Gas Savings
    - Findings that can improve the gas efficiency of the contracts.
- Informational
    - Findings including recommendations and best practices.

Low Findings

1. Low - Some variables are being shadowed

Proof of Concept

Following local variables are being shadowed in the contracts:

- GatedDistributor.constructor(address,uint128,uint256,string,uint96,address,address,bytes32,uint256).guildId (contracts/GatedDistributor.sol#37) shadows:
    - - RequestGuildRole.guildId (contracts/RequestGuildRole.sol#41) (state variable)
- GatedDistributor.constructor(address,uint128,uint256,string,uint96,address,address,bytes32,uint256).jobId (contracts/GatedDistributor.sol#41) shadows:
    - - RequestGuildRole.jobId (contracts/RequestGuildRole.sol#38) (state variable)
- GatedDistributor.constructor(address,uint128,uint256,string,uint96,address,address,bytes32,uint256).oracleFee (contracts/GatedDistributor.sol#42) shadows:
    - - RequestGuildRole.oracleFee (contracts/RequestGuildRole.sol#35) (state variable)

Impact

Variable shadowing can cause unnecessary confusion for the reviewer / tester. Sometimes the confusion can cause mistaken use of the variable which can end up in an error.

Recommendation

Rename the local variables that shadow another component.

2. Low - Some functions use Timestamp for comparison

Proof of Concept

Following functions use timestamps for comparison:

- GatedDistributor.claim() (contracts/GatedDistributor.sol#59-67) uses timestamp for comparisons
    - Dangerous comparisons:
    - - block.timestamp > distributionEnd (contracts/GatedDistributor.sol#60)
- GatedDistributor.withdraw(address) (contracts/GatedDistributor.sol#90-97) uses timestamp for comparisons
    - Dangerous comparisons:
    - - block.timestamp <= distributionEnd (contracts/GatedDistributor.sol#91)

Impact

Use of timestamps for comparison can be manipulated by miners. For example, Bob's contract relies on block.timestamp for its randomness. Eve is a miner and manipulates block.timestamp to exploit Bob's contract.

Recommendation

Avoid relying on block.timestamp.

Informational Findings

3. Informational - Pragma version^0.8.17 necessitates a version too recent to be trusted.

Proof of Concept

Pragma version^0.8.17 necessitates a version too recent to be trusted in the following contracts:

- GatedDistributor.sol
- GatedERC721.sol
- RequestGuildRole.sol
-

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.17;
```

Impact

Using a latest, relatively untrusted version of Solidity can produce undesired side effects.

Recommendation

Consider deploying with:

- 0.6.12
- 0.7.6
- 0.8.7