GGP Vault Audit

AlwaysBeGrowing

Reviewer

Kyle Trusler | Namaskar

February 19, 2024

1 Executive Summary

Over the course of one week, SeaFi engaged with Always Be Growing to review GGP Vault.

A total of 3 issues were found.

Repository	Commit
GGP Vault	93b1f825014c08117194204a292ec440dfd52f1b

Summary

Type of Project	Smart Contract
Timeline	Feb 10, 2024 - Feb 20, 2024
Methods	Manual Review
Documentation	High
Testing Coverage	Full Coverage

Total Issues

Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	2
Gas Optimizations and Informational	1

Contents

2 Always Be Growing

ABG is a talented group of engineers focused on growing the web3 ecosystem. Learn more at https://abg.garden

3 Introduction

The following is a security review of the GGP Vault contract.

The GGP Vault contract is an ERC4626 compliant vault with the GoGoPool protocol token, GGP, as an underlying asset. The funds are utilized in a strategy of staking GGP tokens on behalf of trusted node operators and receiving the rewards generated as yield.

Upon deposit of GGP, a number of ERC20 receipt tokens of xGGP will be minted. These tokens represent the share of assets held within the contract (referred to henceforth as the "vault"). The vault will use the GGP tokens to generate yield. The strategy for generating yield involves staking the GGP tokens on behalf of trusted node operators participating in the GoGoPool protocol. The GoGoPool protocol provides rewards for staking, and those will be accrued by the vault, and increase the value of the xGGP tokens. The xGGP tokens can be redeemed for the underlying GGP tokens.

Disclaimer: This security review does not guarantee against a hack. It is a snapshot in time of brink according to the specific commit by a single human. Any modifications to the code will require a new security review.

4 Architecture

The vault inherits much of its base functionality from trusted and audited contracts from the OpenZeppelin contract library. However, the vault has been modified to incorporate features for added security and the ability to interact with the GoGoPool protocol. The vault has been designed to be upgradable, allowing for future improvements and bug fixes.

The areas of focus therefore are:

1. Does the contract adhere to the ERC-4626 standard?
2. Do the security features work as intended?

3. Are the changes to the OpenZeppelin contracts secure?

5 Methodology

The following are steps I took to evaluate the system:

- Clone & Setup project
- Read Readme and related documentation
- Line by line review
  - Contract
  - Interface
  - Unit Tests
  - Integration Tests
- Review Deployment Strategy
- Review Operation Strategy
- Run Slither
- Run ERC-4626 Solmate Test Suite

6 Findings

6.1 Critical Risk

No findings

6.2 High Risk

No findings

6.3 Medium Risk

No findings

6.4 Low Risk

2 findings

6.4.1 Unprotected upgradable contract

Severity: Medium

Context: GGPVault.sol

Description: While the initializer function is protected by the `initializer` modifier, preventing the re-initialization of the contract, the implementation contract has not been initialized. This means that the contract can be initialized by anyone, potentially causing unexpected behavior.

Recommendation: The implementation contract should be initialized in the constructor of the proxy contract. This will prevent the contract from being initialized by anyone other than the initializer of the contract.

```
constructor(){
    _disableInitizliers();
}
```

Resolution: The recommended code was added to the contract in commit 9ebc70d.

6.4.2 Lack of input validation

Severity: Low

Context: GGPVault.sol#L81

Description: The `setAssetCap` (and `setTargetAPR` in a future version) functions do not validate input beyond the access control checks. While not directly a vulnerability, improper setting of these values could disrupt the contract's economics or functionality.

Recommendation: Add input validation to the `setGGPCap` and `setTargetAPR` functions to ensure that the inputs are within the expected range.

Resolution: The SeaFi team has acknowledged the finding and will be careful with the setting of these values, aiming to add this feature in the future version of the contract.

6.5 Informational

1 finding

6.5.1 Shadowing Variables

Severity: Informational

Context: GGPVault.sol#L140

Description:

The vault inherits from the `OwnableUpgradeable` contract, which has a function `owner`. It is good practice to avoid using the same name for newly declared variables, as it can lead to confusion and unexpected behavior. This is not a security risk for the vault with this particular code, but it can be confusing to developers and may lead to unexpected behavior.

```
function maxRedeem(address owner) public view override returns (uint256) {
    // `owner` here shadows the `owner` function from OwnableUpgradeable
}
```

Recommendation: Rename the variable to something that does not shadow the `owner` variable from `OwnableUpgradeable`. I recommend `shareHolder` as it is more descriptive of the variable's purpose.

```
- function maxRedeem(address owner)
+ function maxRedeem(address shareHolder)
```

Resolution The instances of the `owner` variable have been changed to `shareHolder` in commit cc704a5

7 Additional Risks

There are some additional risks. While they are not directly related to the code, they are still important to consider.

7.1 GGP Price fluctuations

Given the volatile price of the GGP token, the vault will ultimately be exposed to those price changes. The direct consequence of these changes can be seen in the node's collateralization ratio. If the price drops significantly, the node will be under collateralized and will receive the required rewards, however the GGP Cap will be artificially low, and overall have reward inefficiency. On the other hand, if the price increases significantly, the node will be overly collateralized, the GGP Cap will be too high, and it will accrue more in rewards than in actuality. This delta could potentially lead to insufficient liquidity in the vault.

7.2 Upgradable Contracts

7.2.1 GGP Vault

The vault itself is upgradable. This introduces a risk of the contract changing at any time by the owner. Additionally, there is added complexity, which could lead to a faulty upgrade. This process can be mitigated by having a robust upgrade process, including a well-tested contract before the upgrade.

7.2.2 GoGoPool Contracts

This contract interacts with the GoGoPool protocol. The GoGoPool protocol is also upgradable. This introduces a risk of the contract changing at any time by the owner. As a result, the contracts that are being interacted with: the storage and the staking contracts, should be monitored by the team for changes.

7.3 External Contract Interaction

7.3.1 GGP Token

During normal operations of the vault, the transfers and approvals to the GGP token are done safely. The GGP token has been audited and follows the ERC20 standard.

7.3.2 Staking Contract

The vault interacts with the GoGoPool protocol.

In `GGPVault.stakeAndDistributeRewards`, the external function call `stakingContract.stakeGGPOnBehalfOf`, calls the GoGoPool protocol. This is a trusted

contract without a risk of re-entry. However, as mentioned, the GoGoPool protocol is upgradable. This introduces a risk of the contract changing at any time by the owner. Due to this, the contracts that are being interacted with--the storage and the staking contracts--should be monitored by the team for changes.

7.4 Centralization Risk

7.4.1 Elevated Privileges

The owner of the contract as well as the trusted node operators have a large amount of control over the vault. The owner can upgrade the contract, and the trusted node operators can change the configuration of the vault. This introduces a risk of centralization. The team should consider ways to mitigate this risk for the long-term health of the vault.

7.4.2 Role Setting

Given the power of the elevated roles, the setting of new node operators or changing the owner must be done with caution.

7.5 Other Misc

The Solidity version ^0.8.20 used in the vault is potentially too new to be fully trusted.

Test coverage is complete.

The contract is well documented with NatSpec and the documentation website.

7.6 Conclusion

The findings were of low and informational severity. The low-severity findings were a potential griefing in the implementation contract and the unbound variable setting. The informational severity finding was a shadowed variable. There are additional risks to consider, such as the GGP price fluctuations, the upgradability of the contracts, and the centralization risk.

I believe the answers to the questions set out to be: The contract adheres to the ERC-4626 standard, the security features work as intended, and the changes to the OpenZeppelin contracts are secure.