Smart contract security audit

Gravity

v.1.4

Table of Contents

1.0 Introduction

1.1 Project engagement

During July of 2021, Gravity engaged CTDSec to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. Gravity provided CTDSec with access to their code repository and whitepaper.

1.2 Disclaimer

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the network's fast-paced and rapidly changing environment, we at CTDSec recommend that Gravity team put in place a bug bounty program to encourage further and active analysis of the smart contract.

2.0 Coverage

2.1 Target Code and Revision

For this audit, we performed research, investigation, and review of the Gravity contract followed by issue reporting, along with mitigation and remediation instructions outlined in this report. The following code files are considered in-scope for the review:

Source:

audit-gravity-ctdsec-v1-core-main.zip [SHA256] -

ba1fad52d2890f95b6fd5fb769e5a1593a03ee82ab1ff4c779751049abac1b7a

2.2 Attacks made to the contract

In order to check for the security of the contract, we tested several attacks in order to make sure that the contract is secure and follows best practices.

№	Issue description.	Checking status
1	Compiler warnings.	PASSED
2	Race conditions and Reentrancy. Cross-function race conditions.	PASSED
3	Possible delays in data delivery.	PASSED
4	Oracle calls.	PASSED
5	Front running.	PASSED
6	Timestamp dependence.	PASSED
7	Integer Overflow and Underflow.	PASSED
8	DoS with Revert.	PASSED
9	DoS with block gas limit.	LOW ISSUES
10	Methods execution permissions.	PASSED
11	Economy model. If application logic is based on an incorrect economic model, the application would not function correctly and participants would incur financial losses. This type of issue is most often found in bonus rewards systems, Staking and Farming contracts, Vault and Vesting contracts, etc.	PASSED
12	The impact of the exchange rate on the logic.	PASSED
13	Private user data leaks.	PASSED
14	Malicious Event log.	PASSED
15	Scoping and Declarations.	PASSED
16	Uninitialized storage pointers.	PASSED
17	Arithmetic accuracy.	PASSED

18	Design Logic.	PASSED
19	Cross-function race conditions.	PASSED
20	Safe Zeppelin module.	PASSED
21	Fallback function security.	PASSED
22	Overpowered functions / Owner privileges	PASSED

3.0 Security Issues

3.1 High severity issues [0]

No high severity issues found.

3.2 Medium severity issues [0]

No medium severity issues found.

3.3 Low severity issues [1]

1. Out of gas Issue:

Function updateWithdrawableFee() uses the loop to update withdrawableFee of all users.

Function will be aborted with OUT_OF_GAS exception if there is a long users list.

Recommendation:

Check that the user array length is not too big.

4.0 Owner Privileges

WalletTimeLock (no access to user funds):

- ❖ Owner can withdraw any ERC20 tokens.

<u>Dev update:</u>

This contract is used to enforce a 7-day timelock on Gravity project wallets, i.e. unsold IDO tokens, unallocated Farm Rewards etc.

Any function that is called on a locked wallet will emit an event on-chain. The event will show information about the function being called, i.e. token transfers, wallet/contract ownership changes etc.

These events can be monitored on-chain by users and any event can be verified before the 7-day timelock allows the function to be completed. Users can automate the monitoring process by using Open Zeppelin "Sentinels" which can be set up to send alerts to users through social media or Telegram etc. We will have a section in our Gitbook that explains the use of Sentinels and encourages users to make their own.

Example: If a "transfer" event is emitted, users would see a transfer from a wallet to a new address. This event would specify info such as "Transfer, Number of GFI tokens, From wallet, to New Address".

Users can then check the new address on-chain to verify where the GFI tokens are being sent and for what reason (such as sending GFI Rewards to a new farm contract).

VestingV2 (no access to user funds):

- ❖ Owner can withdraw all funds after 30 days from LockEnd time.
- ❖ Owner can change:
  - ➢ Governor address.
  - ➢ Stop fee collection status.
  - ➢ Callers share.
  - ➢ Users list members.

<u>Dev update:</u>

VestingV2 is a contract that holds some early investor GFI tokens. This contract is not used by users of the Gravity Finance project through any of the platform products.

Owner “withdraw all” function is a fallback function in the event there are issues with users claiming GFI or wETH.

The other functions mentioned are required for adding new users to the contract and to set up wETH fee collection.

Locking (no access to user funds):

- ❖ Owner can withdraw all funds after 30 days from LockEnd time.
- ❖ Owner can change:
    - ➢ Governor address.
    - ➢ Stop fee collection status.
    - ➢ Users list members.

<u>Dev update:</u>

This contract is the locking functionality for the Treasury-owned GFI tokens, Seed Investors, Team Members, Founders etc. These tokens are locked for 12 months from the deployment date.

Owner “withdraw all” is uncallable.

TransferFrom was used instead of transfer, so the Locking contract would need to approve itself to spend its tokens (which is not implemented).

The other functions mentioned are required for adding new users to the contract and to set up wETH fee collection.

IOUToken (no access to user funds):

- ❖ Owner can mint and burn.

<u>Dev update:</u>

This contract is part of the original Gravity IDO platform and is deprecated.

This contract was deployed by the IDO contract, so the owner will always be the

IDO (which was required for the IDO to be able to mint and burn tokens during the IDO process).

Incinerator (no access to user funds):

- ❖ Owner can change the slippage value.

- Owner can convert earnings to GFI and burn them.

<u>Dev update:</u>

This contract is used to swap wETH to GFI and burn GFI tokens from the total supply without making a claim on the wBTC backing from the Governance Contract.

The slippage value can be changed to handle an event where GFI liquidity might be low and cause issues swapping from wETH to GFI.

The main use case of this function is for some pools that hold GFI. These pools can claim the wETH earned against the GFI, use it to buy and burn GFI, benefiting all GFI holders.

Anyone can convert earnings to GFI and burn them, but the owner is allowed to do this without using the price oracle. This is a special owner privilege because without it, callers could make the Incinerator make bad trades.

GravityToken (no access to user funds, but setting governance forwarding to true, and setting governance address to a non governance contract would make all token transfers fail):

- Owner can change:
  - Governance address.
  - Governance forwarding status.

<u>Dev update:</u>

Owner functions are needed to set up platform fee distribution to token holders.

GravityIDO:

- Owner can change GFI/WETH addresses.
- Owner can withdraw.

<u>Dev update:</u>

This was the original Gravity IDO contract which is now deprecated.

Governance (no access to user funds, but access to wETH and wBTC backing):

- Owner can change tiers.

Dev update:

Tiers will be reviewed periodically and can be updated in the event the market price of GFI changes drastically. We feel it is unrealistic to expect users to buy enough GFI for tier 3 if the price to do so has skyrocketed.

The governance contract is upgradeable, allowing new functions to be added or updated, such as tier logic, fee distribution, and investment strategies for idle assets in the governance contract.

As this is a powerful feature it needs to be mitigated in order to protect user funds and project backing.

Short term, the upgrades will be on a timelock, allowing users to monitor the smart contract for any changes (using Sentinels).

Long term, GFI holders will be able to vote on upgrades as GFI shifts to a DAO.

Flat_GFIFarm, Flat_FarmV1 (access to user funds):

- ❖ Owner can initialize the contract.
- ❖ Owner can withdraw rewards.

Dev update:

These are the legacy contracts that are already deprecated and will be phased out for farm V2 contracts.

FarmTimeLock (access to user funds, since the farmV1 contract has access to user funds):

- ❖ Owner can initiate withdraw rewards.
- ❖ Owner can withdraw any ERC20 token.

Dev update:

This contract was created to implement a 7-day timelock on functions being called to the V1 farm contracts.

Owner can only withdraw rewards and user-funds if the deposit token is the same as the reward token. If user funds are different to the reward token then the owner cannot withdraw user-funds.

All functions are behind a 7-day timelock, so any function calls will emit an event on-chain that users can monitor and check (using Sentinels).

V1 farms are deprecated.

FarmFactory (access to user funds through ‘harvest fee’ which is capped at 5%):

- ❖ Owner can change:
  - ➢ Whitelist members.
  - ➢ Harvest fee.
  - ➢ Incinerator address.
  - ➢ Fee manager address.
  - ➢ Farm valid status.

<u>Dev update:</u>

User-deposited funds cannot be accessed. Only rewards can be “taxed” as a performance fee and only when the owner is a smart contract (not a regular wallet).

Whitelist members can be updated to allow specific addresses to not pay a performance fee.

Harvest fee (performance fee) can be set to a maximum of 5%. The performance fee is taken at the time of harvesting rewards.

Incinerator address is used to set the address of the incinerator so that wETH earnings can be sent to it.

Fee Manager address is used to update where to send performance fees if the reward token is not GFI. If the reward is GFI the GFI is burned.

Farm valid status controls what farms are created, but it is done in such a way that 3rd parties do not need to trust us with their tokens for new farms, rather they trust the Smart Contract.

CompounderFactory (access to user funds through ‘harvest fee’ which is capped at 5%):

- ❖ Owner can change:
  - ➢ Vault fee.
  - ➢ Tier checker address.
  - ➢ Check tiers status.
  - ➢ Max caller reward min harvest and caller fee percent of share info item.
  - ➢ Shared variables.

Dev update:

User-deposited funds cannot be accessed. Only rewards can be “taxed” as a performance fee.

Vault fee (performance fee) can be set to a maximum of 5%. The performance fee is taken at the time of harvesting rewards.

Tier checking variables are features for future contracts that will calculate a users tier based on their total amount of GFI.

Max Caller reward, min harvest, and caller fee percent are all variables we will use to tune the payout for compounding the farms. If the farm is being compounded too much or the reward is too high we can adjust the above variables to fix that.

5.0 Summary of the audit

Smart contracts contain low issues and it’s safe to deploy. All owner privileges where well detailed by development team and confirmed by CTDsec.