gTrade Security Review

Pashov Audit Group

Conducted by: pashov

September 6th, 2023

Contents

1. About pashov

Krum Pashov, or **pashov**, is an independent smart contract security researcher. Having found numerous security vulnerabilities in various protocols, he does his best to contribute to the blockchain ecosystem and its protocols by putting time and effort into security research & reviews. Check his previous work here or reach out on Twitter @pashovkrum.

2. Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where I try to find as many vulnerabilities as possible. I can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

3. Introduction

A time-boxed security review of the **gTrade** protocol was done by **pashov**, with a focus on the security aspects of the application's smart contracts implementation.

4. About gTrade

The `gTrade` protocol is a leveraged trading platform developed by Gains Network. It allows for up to 150x on crypto tokens, 1000x on forex and 250x on commodities. The architecture mainly revolves around the `GNS` ERC20 token and ERC721 utility tokens, both of which provide different utility when using the platform - reduced spread, governance and others.

The protocol added two new features:

- Staking, which allows users to stake `GNS` and receive `DAI` rewards from trading activity on the platform
- Compensation Handler, which is currently used to compensate holders of the utility ERC721 tokens, because they are getting deprecated

The Compensation Handler will also compensate the dev fund because the deprecated ERC721 tokens were a source of revenue to the fund up until now.

Observations

Anyone holding the `MINTER_ROLE` can mint `GNS` tokens freely.

The dev fund compensation can be executed only on the Arbitrum network.

There is no lock on staking, users can stake and unstake in the same block.

The reward "distribution" happens on a per-trade (open, close, liquidation) basis. Because of this, distribution can't be timed as it can happen many times in a minute or zero times in hours.

Privileged Roles & Actors

- `GNS`'s `MINTER_ROLE` - holder of the role can freely mint `GNS` tokens to any address, the `CompensationHandler` contract will have this role
- Unlock Manager - can create revocable unlock schedules
- Governance - the governance account address, can call `scheduleDevFundUnlock` to start the cliff vesting for dev fund compensation, create/revoke revocable unlock schedules and set `unlockManagers`
- User - can create non-revocable unlock schedules for himself

5. Risk Classification

Severity	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

5.1. Impact

- High - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- Medium - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- Low - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

5.2. Likelihood

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

5.3. Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

6. Security Assessment Summary

review commit hash **- 5dfd04b320b6f14ccbabfe2b790987826422b545**

fixes review commit hash **- cf0e7aae00542b479e34ed7a0fbc77c9b0c3084e**

Scope

The following smart contracts were in scope of the audit:

- `GNSCompensationHandlerV6_4_1`
- `GNSStakingV6_4_1`

7. Executive Summary

Over the course of the security review, pashov engaged with gTrade to review gTrade. In this period of time a total of **3** issues were uncovered.

Protocol Summary

Protocol Name	gTrade
Date	September 6th, 2023

Findings Count

Severity	**Amount**
Medium	1
Low	2
Total Findings	**3**

Summary of Findings

ID	**Title**	**Severity**	**Status**
[M-01]	Insufficient input validation	Medium	Resolved
[L-01]	Implementation contract can be initialized	Low	Resolved
[L-02]	Use a two-step access control transfer pattern	Low	Resolved

8. Findings

8.1. Medium Findings

[M-01] Insufficient input validation

Severity

Impact: High, as it can lead to stuck funds

Likelihood: Low, as it requires a bad user error

Description

In `GNSStakingV6_4_1::createUnlockSchedule` we have the `UnlockScheduleInput calldata _input` parameter, where most of the fields in the struct are properly validated to be in range of valid values. The issue is that the `start` field of the `UnlockScheduleInput` is not sufficiently validated, as it can be too further away in the future - for example 50 years in the future, due to a user error when choosing the timestamp. This would result in (almost) permanent lock of the `GNS` funds sent to the method.

Recommendations

Add a validation that the `start` field is not too further away in the future, for example it should be max 1 year in the future.

8.2. Low Findings

[L-01] Implementation contract can be initialized

The `GNSStakingV6_4_1` is an implementation contract that is expected to be used through a proxy. Since implementation contracts shouldn't be used, it is a convention to disallow their initialization. Consider adding an empty constructor that calls `_disableInitializers()` in `GNSStakingV6_4_1`.

[L-02] Use a two-step access control transfer pattern

The `GNSStakingV6_4_1::setGovFund` uses a single-step access control transfer pattern. This means that if the current `govFund` account calls `setGovFund` with an incorrect address, then this `govFund` role will be lost forever along with all the functionality that depends on it. Follow the pattern from OpenZeppelin's Ownable2Step and implement a two-step transfer pattern for the action.