EASE AUDIT
GVTOKEN

Mail
contact@ease.org

August 15th, 2022

Lead Auditor
Robert M.C. Forster

Table of Contents

1. Summary

This audit was executed on Ease's gvToken system over a period of 1 week with a team of 1 engineer. Only 1 engineer was available internally that had not had prior exposure to the code being developed.

The commit audited:

- **9379907aaea908c0cc762310f90e16b95f2ab1ab**

The contracts in-scope are:

- BribePot.sol
- gvToken.sol
- TokenSwap.sol
- EaseToken.sol

Audit Process

1. Read documentation, read and understand the contracts.
2. Converse with developers to ensure correct understanding and ask any questions.
3. Identify potential attack vectors.
4. Use automated tests for an initial overview and to catch any obvious problems.
5. Analyze unit tests for any missing edge cases.
6. Individual manual auditing.
7. Collaborate on findings.
8. Finish manual auditing collaboratively.
9. Return findings to client.
10. Review client changes and finalize audit.

2. Terminology

We generally follow Immunefi's Vulnerability Severity Classification System:

1. Note
- Not following best practices

2. Low
- Contract fails to deliver promised returns, but doesn't lose value
- Loss of value in small amounts

3. Medium
- Contract out of gas
- Contract consumes unbounded gas
- Denial of service

4. High
- Theft of yield
- Token holders temporarily unable to transfer holdings
- Freeze entire upgradeable contract's operation

5. Critical
- Empty or freeze the contract’s holdings (e.g. economic attacks, flash loans, reentrancy, logic errors, integer over-/under-flow)
- Freeze entire non-upgradeable contract's operation

3. Overview of Findings

Resolved Findings	
Critical	1
High	1
Medium	1
Low	0

Acknowledged Findings	
Critical	0
High	0
Medium	0
Low	1

4. Resolved Findings

1. Delegated amounts may not change on withdrawal

Background: The gvToken contract gives users $gvEASE tokens for voting. Users must then delegate these votes to themself or others for them to be used for voting.

Critical

Explanation: The problem is that when a withdrawal from the gvToken contract occurs, the delegated balance of tokens is not lowered according to the withdrawal. This results in a user being able to deposit tokens, delegate, then withdraw, and keep their original votes. This could be exploited by looping this activity over and over (although there is a 7 day period they must wait each time) to add many fake votes.

Rating: This bug would have allowed users to multiply their weight in governance infinitely and could fairly easily be used to take over governance before being noticed, so it's been given a critical rating.

Resolution: This line to subtract from delegate balance upon withdrawal was added.

https://github.com/EaseDeFi/gvToken/blob/7a5e12e68748627412dd2aefba9c81ebc39520e8/contracts/core/GvToken.sol#L226

4. Resolved Findings

2. User may withdraw without subtracting from bribe balance

Background: Instead of staking, a user may allow their tokens to be bribed by others. To do this, they call the ***** function to deposit stake into the BribePot contract. Their balance on this contract is used as a balance to gain rewards from bribes with an SNX-style rewards contract. When tokens are withdrawn from the gvToken contract, they should no longer be able to be bribed.

Explanation: The contract allows a user to withdraw their tokens without subtracting from their BribePot balance if they pass `includeBribePot` as false.

High

https://github.com/EaseDeFi/gvToken/blob/9379907aaea908c0cc762310f90e16b95f2ab1ab/contracts/core/GvToken.sol#L303

This allows a user to deposit tokens, enter them into the BribePot, then withdraw them but keep a balance. A user may keep doing this to build their balance to unreasonable amounts and steal rewards, although each withdrawal takes 7 days.

Rating: This vulnerability is given a "High" rating because it is easy to pull off and could lead to bribing no longer working, however the upgradeability of this contract would mean it would be relatively easy to fix without long-term damage.

Resolution: The contract now automatically withdraws with this line:

https://github.com/EaseDeFi/gvToken/blob/7a5e12e68748627412dd2aefba9c81ebc39520e8/contracts/core/GvToken.sol#L571

4. Resolved Findings

3. BribePot rewards may be stopped by manipulating periodFinish

Background: The BribePot has a set time to finish distributing rewards. After this finish, no more rewards will be distributed to users. The `periodFinish` variable keeps track of this and must be the last week that any rewards are distributed.

Explanation: `periodFinish` can be set to next week regardless of other expiries because it does not check other weeks between that time. A malicious actor can start a bribe to finish in one year, cancel, and periodFinish is set to next week regardless of weeks after that.

Medium

https://github.com/EaseDeFi/gvToken/blob/9379907aaea908c0cc762310f90e16b95f2ab1ab/contracts/core/BribePot.sol#L260

Rating: this vulnerability is given a "Medium" rating because it is easy to pull off and could lead to rewards temporarily not working, however it can be fixed by depositing again and adjusting `periodFinish`.

Resolution: This was resolved with a loop that loops backwards through weeks to find the next-latest bribe expiry and sets periodFinish to that week.

https://github.com/EaseDeFi/gvToken/blob/7a5e12e68748627412dd2aefba9c81ebc39520e8/contracts/core/BribePot.sol#L238

4. Resolved Findings

4. Re-entrancy best practices

Note

Background: Multiple places throughout the contracts do not follow re-entrancy best practices of saving state changes on the contract before transferring tokens to the user.

https://github.com/EaseDeFi/gvToken/blob/9379907aaea908c0cc762310f90e16b95f2ab1ab/contracts/core/GvToken.sol#L270

Rating: This was given a "Note" rating because we could not find any way to exploit it, but it does not follow best practices.

Resolution: Ordering adjusted.

5. Add comments

Note

Background: There's a lack of comments in the code, leading to difficulty in understanding and auditing it.

Rating: This was given a "Note" rating because it affects nothing about the contract but only makes it easier to understand.

Resolution: Many comments added to make the code much more readable.

5. Acknowledged Findings

1. Rewards can get stuck if the venal pot is emptied

Low

Background: The venal pot is the funds that are currently "up for" bribe, a.k.a. the $gvEASE owned by users that others can pay to stake. Rewards are paid weekly for the funds in the venal pot.

Explanation: If all funds from the pot are withdrawn, any ongoing rewards will not be given to anyone and will end up stuck in the contract. This is because users paying to bribe tokens will essentially be paying for 0 tokens and any newcomers will not retroactively receive rewards from this amount.

Rating: This is given a `low` rating because it only affects small amounts of funds and very rarely.

Resolution: Developers decided not to adjust the contract because fixing the relatively harmless bug would introduce extra complication.

5. Acknowledged Findings

2. Unprotected transferFroms

Note

Background: Sometimes the `transferFrom` functions of a token may not function in the way expected. For example, some tokens return `false` instead of erroring when the transfer does not succeed. This means that relying on the transaction to fail for unknown tokens could result in a contract thinking a transaction succeeded when it really failed.

Explanation: Within TokenSwap.sol there are a couple places where the `transferFrom` call is not protected:

https://github.com/EaseDeFi/gvToken/blob/9379907aaea908c0cc762310f90e16b95f2ab1ab/contracts/core/TokenSwap.sol#L26

Rating: This is a note because we cannot find any way it could be exploited and is only best practice.

Resolution: Developers decided not to adjust the contract because the tokens will always be known and no harm can be done.

6. Limitations and Use

- The purpose of this audit (report) is to cover security threats, commonly referred to as "hacking", within the general community.
- This report is not an endorsement of the business and its practices.
- This report only covers the submitted smart contract code and is meant to be a tool for the discovery of overlooked vulnerabilities during the initial development time of the aforementioned smart contract code.
- A report cannot discover all vulnerabilities that may exist. Do not take this report as an endorsement of the safety or security of the submitted smart contract code.
- This report was conducted with the provided specifications from the business. Any vulnerabilities that are the result of parameters beyond the submitted smart contract code, such as the blockchain, its programming language, underlying third-party infrastructure, and off-chain (front-end or back-end) code are not within the scope of this audit.
- If the submitted smart contract code is modified or updated at a future point in time from the original submission, this audit may no longer be relevant.
- This report was originally written in English. Any errors that are the result of translation are not the responsibility of the authors of this report.