HACKEN

# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

**Customer**: Gogo Protocol
**Date**: February 14th, 2022

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io



## Document

| | |
|---|---|
| **Name** | Smart Contract Code Review and Security Analysis Report for Gogo Protocol. |
| **Approved by** | Andrew Matiukhin \| CTO Hacken OU |
| **Type** | Governance Staking |
| **Platform** | Polygon / Solidity |
| **Methods** | Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review |
| **Repository** | https://github.com/gogocoin/gogo-contracts |
| **Commit** | 0204c6d60f7aaa5c195573fbd0e8388c9bc4a2ef |
| **Deployed contract** | - |
| **Technical Documentation** | NO |
| **JS tests** | YES |
| **Website** | |
| **Timeline** | 17 DECEMBER 2021 - 14 FEBRUARY 2022 |
| **Changelog** | 24 DECEMBER 2021 - Initial Audit<br>14 FEBRUARY 2022 - Second Review |

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Introduction

Hacken OÜ (Consultant) was contracted by Gogo Protocol (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contract and its code review conducted between December 17th, 2021 - February 14th, 2022.

# Scope

The scope of the project is smart contracts in the repository:

**Repository:**
https://github.com/gogocoin/gogo-contracts
**Commit:**
0204c6d60f7aaa5c195573fbd0e8388c9bc4a2ef
**Technical Documentation:** No
**JS tests:** Yes (included: test/govstaking.test.ts)
**Contracts:**
gov/CommunityRewards.sol
gov/CommunityRewardsManager.sol
gov/GovStaking.sol
gov/GovStakingStorage.sol
gov/RewardsDistributionRecipient.sol
gov/CommunityRewardsV2.sol
gov/GovStakingv2.sol

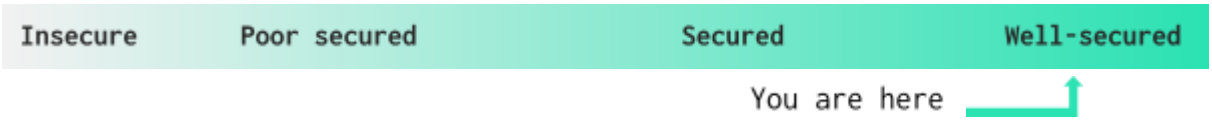
We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

| Category | Check Item |
|---|---|
| Code review | ▪ Reentrancy<br>▪ Ownership Takeover<br>▪ Timestamp Dependence<br>▪ Gas Limit and Loops<br>▪ DoS with (Unexpected) Throw<br>▪ DoS with Block Gas Limit<br>▪ Transaction-Ordering Dependence<br>▪ Style guide violation<br>▪ Costly Loop<br>▪ ERC20 API violation<br>▪ Unchecked external call<br>▪ Unchecked math<br>▪ Unsafe type inference<br>▪ Implicit visibility level<br>▪ Deployment Consistency<br>▪ Repository Consistency<br>▪ Data Consistency |

| | |
|---|---|
| Functional review | ▪ Business Logics Review<br>▪ Functionality Checks<br>▪ Access Control & Authorization<br>▪ Escrow manipulation<br>▪ Token Supply manipulation<br>▪ Assets integrity<br>▪ User Balances manipulation<br>▪ Data Consistency manipulation<br>▪ Kill-Switch Mechanism<br>▪ Operation Trails & Event Generation |

## Executive Summary

According to the assessment,the Customer's smart contracts are well-secured.




Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview section.

As a result of the audit, security engineers found **1** medium and **5** low severity issues.

After the second review security engineers found two new contracts (with prefixes V2) and fixes that were implemented in new contracts. Therefore **2** low severity issues still persist.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Severity Definitions

| Risk Level | Description |
|---|---|
| Critical | Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations. |
| High | High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions |
| Medium | Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations. |
| Low | Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution |

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Audit overview

## Critical

No critical issues were found.

## High

No high severity issues were found.

## Medium

1. Possible inconsistency.

   While both contracts GovStaking and CommunityRewards are using GovStakingStorage and both rely on the data from the storage, it's very important that all they have the same storage instance. If the storage address will be different, there will be inconsistency in the state which could lead to unpredictable results.

**Contract**: GovStaking.sol, CommunityRewards.sol, CommunityRewardsManager.sol

**State Variable**: store

**Recommendation**: Please use the manager to ensure that Staking and Rewards contracts have the same store address.

**Status**: Fixed in V2 contracts

## Low

1. Unused import statement.

**Contract**: GovStakingStorage.sol

**Import**: hardhat/console.sol

**Recommendation**: Remove unused import statement.

**Status**: Not fixed

2. Incorrect contract imported.

   CommunityRewards contract inherits RewardsDistributionRecipient but it imports that from the "staking" directory of the project instead of the "gov" one. Since those contracts are the same it's not a big deal, but for consistency meaning it's better to import the correct contract.

**Contract**: CommunityRewards.sol

**Recommendation**: Please fix the import statement.

**Status**: Fixed in V2 contract

3. Too many digits.

   Literals with many digits are difficult to read and review. It's always a better idea to use solidity provided time unit suffixes like:

   - 1 == 1 seconds
   - 1 minutes == 60 seconds
   - 1 hours == 60 minutes
   - 1 days == 24 hours
   - 1 weeks == 7 days

**Contract**: GovStaking.sol

**Function**: constructor

**Recommendation**: Please use time unit suffixes.

**Status**: Not fixed.

4. No events emitted.

   Changing the contract values, which are critical, is always recommended to follow with the events so the community members are always able to track such changes off-chain.

**Contract**: CommunityRewards.sol

**Functions**: setEarlyExitFee

**Recommendation**: Please emit events on fees changed.

**Status**: Fixed.

5. A public function that could be declared external.

   **public** functions that are never called by the contract should be declared **external** to save gas.

**Contracts**: GovStaking.sol, GovStakingStorage.sol, CommunityRewards.sol

**Functions**: GovStaking.claim, GovStaking.claimAll, GovStaking.earned, GovStaking.pause, GovStaking.unpause, GovStakingStorage.getTotalLockedGogo, GovStakingStorage.getTotalRewardRates, GovStakingStorage.emergencyWithdraw, CommunityRewards.getReward, CommunityRewards.getRewardWithLoss,

**Recommendation**: Use the **external** attribute for functions never called from the contract.

**Status**: Fixed.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

## Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

As a result of the audit, security engineers found **1** medium and **5** low severity issues.

After the second review security engineers found two new contracts (with prefixes V2) and fixes that were implemented in new contracts. Therefore **2** low severity issues still persist.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Disclaimers

## Hacken Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

## Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.