HACKEN

# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

**Customer**: Proto Gold DAO
**Date**: August 31th, 2021

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 Proto Gold DAO. |
| **Approved by** | Andrew Matiukhin \| CTO Hacken OU |
| **Type** | BEP20 token upgradeable; Proto Gold Fuel (PROTO), and Proto Gold DAO (LAW), and ProtoSpaceDrop |
| **Platform** | Binance Smart Chain/Solidity |
| **Methods** | Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review |
| **Repository** | https://github.com/Proto-Gold/Contracts/ |
| **Commit** | 02be8a9254db9773479d34f831ad5d1a4b7be5e0 |
| **Technical Documentation** | YES |
| **JS tests** | YES |
| **Timeline** | 11 AUGUST 2021 - 27 AUGUST 2021 |
| **Changelog** | 20 AUGUST 2021 - End of Phase one<br>27 AUGUST 2021 - Second Review |

## Table of contents

## Introduction

Hacken OÜ (Consultant) was hired by Proto Gold DAO (Customer) to conduct Smart Contract Code Reviews and Security Analysis. This report presents the security assessment of the Customer's smart contracts and its code review conducted between August 11th, 2021 - August 27th, 2021.

## Scope

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

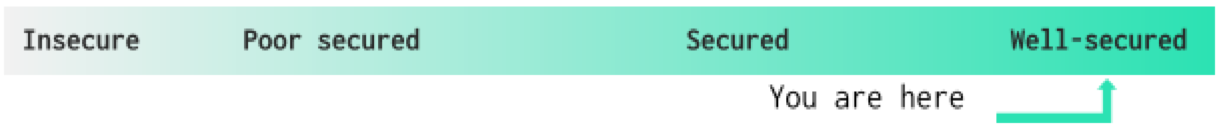
**Repository:**
https://github.com/Proto-Gold/Contracts
**Commit:**
02be8a9254db9773479d34f831ad5d1a4b7be5e0
**Technical Documentation:** Yes
**JS tests:** Yes
**Contracts:**
Contracts-master/contracts/ProtoBEP20V2.sol
Contracts-master/contracts/Proto.gold.v2.sol
Contracts-master/contracts/Pancake.sol
Contracts-master/contracts/ProtoDistribution.sol
Contracts-master/contracts/ProtoDrop.sol
Contracts-master/contracts/ProtoDao.sol
Contracts-master/contracts/ProtoLiquidity.sol
Contracts-master/contracts/ProtoType.sol
Contracts-master/contracts/Address.sol
Contracts-master/contracts/SafeMath.sol
Contracts-master/contracts/ProtoLock.sol
Contracts-master/contracts/IBEP20.sol
Contracts-master/contracts/IProtoDistribution.sol
Contracts-master/contracts/CrowdContributionInterface.sol
Contracts-master/contracts/Proto.gold.v2.1.sol
Contracts-master/contracts/ILawGovernance.sol
Contracts-master/contracts/space-drop/SpaceDrop.sol
Contracts-master/contracts/space-drop/ProtoSpaceDrop.sol
Contracts-master/contracts/space-drop/SpaceDropType.sol
Contracts-master/contracts/law/LawType.sol
Contracts-master/contracts/law/SpaceDropLaw.sol
Contracts-master/contracts/law/LawDrop.sol
Contracts-master/contracts/law/Proto.Law.v2.sol
Contracts-master/contracts/law/LawGovernance.sol
Contracts-master/contracts/law/LawBEP20V2.sol
Contracts-master/contracts/law/IProto.sol
Contracts-master/contracts/law/IProtoSpaceDrop.sol
Contracts-master/contracts/law/LawDao.sol
Contracts-master/contracts/law/Proto.Law.v2.1.sol

We have scanned these smart contracts for commonly known and more specific vulnerabilities. Here are some of the widely 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 analyzed code functionality, manual audit, and automated checks with Mythril and Slither. Any issues our security engineers found during automated analysis were manually reviewed, and essential vulnerabilities are presented in the Audit overview section. Any issues found can be located in the Audit overview section.

## 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 primarily outdated, unused, etc., code snippets that can't significantly impact execution. |

## Audit overview

### 🟥🟥🟥🟥 Critical

No critical issues were found.

### 🟨🟨🟨 High

No high severity issues were found.

### 🟩🟩 Medium

No medium severity issues were found.

### 🟦 Low

1. A public function that could be declared external

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

   **Customer’s response**: Nearly all remaining functions are library functions, and libraries use delegatecalls, and delegate calls don't copy data during a function call; instead, they pass a reference, which doesn't consume any extra gas.

## 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 our preliminary audit, security engineers found one low severity issue. However, after reviewing additional documentation from the customer, our security engineers identified ZERO issues.**

## 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 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.