SMART CONTRACT AUDIT

ZOKYO.

May 5, 2021 | v. 1.0

PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges

TECHNICAL SUMMARY

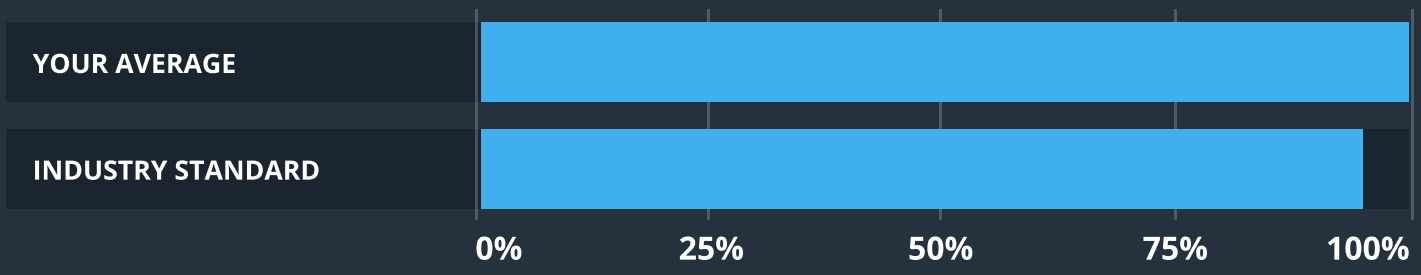
This document outlines the overall security of the Gain DAO smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document the Gain DAO smart contract codebase for quality, security, and correctness.

Contract Status

There were no critical issues found during the audit.

Testable Code



The testable code is 100%, which is above the industry standard of 95%.

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 Ethereum network's fast-paced and rapidly changing environment, we at Zokyo recommend that the Gain DAO team put in place a bug bounty program to encourage further and active analysis of the smart contract.

TABLE OF CONTENTS

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the Gain DAO repository – https://github.com/GainDAO/token/commit/666d27c820896108427c2d8e922cbdfcc2e88387. Commit id – d4aa0b6748a3c643edf5e448444bccd63eee410e.

Throughout the review process, care was taken to ensure that the token contract:

- Implements and adheres to existing Token standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of gas, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of Gain DAO smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1	Due diligence in assessing the overall code quality of the codebase.	3	Testing contract logic against common and uncommon attack vectors.
2	Cross-comparison with other, similar smart contracts by industry leaders.	4	Thorough, manual review of the codebase, line-by-line.

EXECUTIVE SUMMARY

There were no critical issues found during the audit. All the mentioned findings may have an effect only in case of specific conditions performed by the contract owner.

Contracts are well written and structured. The findings during the audit have no impact on contract performance or security, so it is fully production-ready.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the ability of the contract to compile or operate in a significant way.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

MANUAL REVIEW

Token mint emits Paused and Unpaused events, which can be non-intuitive

LOW | RESOLVED

Current mint functionality (GainDAOToken.sol:38) contains a slight hack to allow minting tokens even when the token is paused, which can be non-intuitive. Also, this hack can be removed by slightly modifying the current implementation of _beforeTokenTransfer (GainDAOToken.sol:59).

Recommendation:
The easiest way to remove the need to unpause\pause is to modify _beforeTokenTransfer (GainDAOToken.sol:59), removing whenNotPaused check and instead requiring that contract is unpaused OR from parameter is zero address (which is valid only for mints) effectively allowing mints even when the contract is paused. If for some reason this recommendation isn't desired it's better at least to unpause\pause by directly modifying the underlying variable without emitting events.

Rename parameters of the ERC20Vesting constructor

INFORMATIONAL | RESOLVED

Most of the parameters of ERC20Vesting constructor (ERC20Vesting.sol:51) have the same name as the respective state variable getters, so the compiler raises the "This declaration shadows an existing declaration." warning. This shouldn't affect the functionality in any way, but generally, it's better to minimize compiler warnings.

State variables in ERC20Vesting can be made public

INFORMATIONAL | RESOLVED

Changing state variables to public will auto-generate getters for them (ERC20Vesting.sol:81-121). This doesn't affect compiled bytecode but will reduce the amount of solidity code.

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Secured team

As part of our work assisting Gain DAO in verifying the correctness of their contract code, our team was responsible for writing integration tests using the Truffle testing framework.

Tests were based on the functionality of the code, as well as a review of the Gain DAO contract requirements for details about issuance amounts and how the system handles these.

Contract: ERC20Vesting

- ✓ cannot deploy if grant beneficiary is the zero address (3292ms)
- ✓ cannot deploy if cliff is longer than duration (290ms)
- ✓ cannot deploy if duration is 0 (347ms)
- ✓ cannot deploy if final time is before current time (430ms)
- ✓ should deploy correctly (465ms)
- ✓ cannot release if no tokens are due (325ms)
- ✓ should release if current timestamp more than duration or token revoked (1485ms)
- ✓ should release if current timestamp less than duration or token not revoked (756ms)
- ✓ cannot revoke if not revocable (491ms)
- ✓ cannot revoke if token already revoked (1071ms)
- ✓ should emit event on revoke (2373ms)

Contract: GainDAOToken

- ✓ should deploy with correct name (130ms)
- ✓ should deploy with correct symbol (112ms)
- ✓ should deploy with correct decimals (352ms)
- ✓ should deploy with admin role (293ms)
- ✓ should deploy with minter role (177ms)
- ✓ should deploy with correct cap (61ms)
- ✓ should deploy with pauser role (51ms)
- ✓ should be in pause state after deploy (115ms)
- ✓ cannot unpause if caller does not have the pauser role (294ms)
- ✓ should unpause correctly (484ms)
- ✓ cannot mint if caller does not have the minter role (169ms)
- ✓ should mint tokens correctly (1080ms)

✓ should mint when amount is less than cap (497ms)
✓ cannot mint if amount exceeds the cap (430ms)
✓ beforeTokenTransfer() (527ms)

26 passing (1m)

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts/	100.00	100.00	100.00	100.00	
ERC20Vesting.sol	100.00	100.00	100.00	100.00	
GainDAOToken.sol	100.00	100.00	100.00	100.00	
All files	**100.00**	**100.00**	**100.00**	**100.00**	

We are grateful to have been given the opportunity to work with the Gain DAO team.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

Zokyo's Security Team recommends that the Gain DAO team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.