SMART CONTRACT AUDIT

ZOKYO.

March 15th, 2022| 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

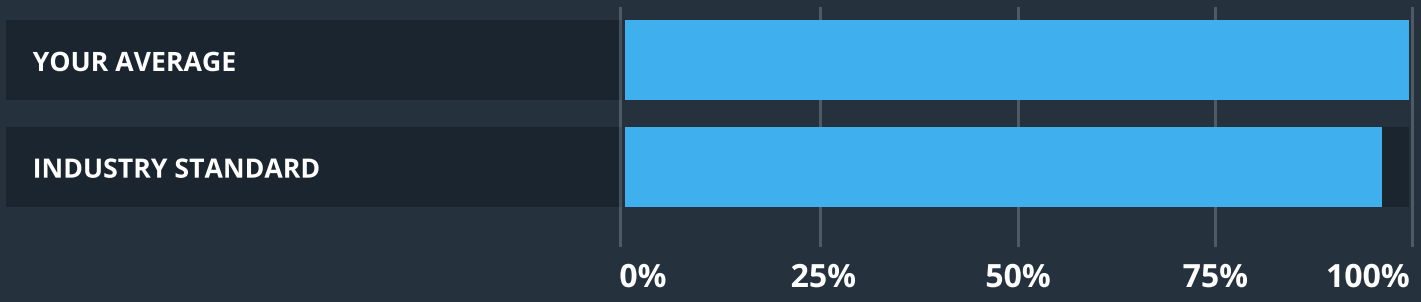
This document outlines the overall security of the Penguin Karts smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document Penguin Karts smart contract codebase for quality, security, and correctness.

Contract Status

There were no critical, high or medium 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 security of the contract we at Zokyo recommend that the Penguin Karts 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 Penguin Karts smart contract's source code was taken from the smart contract provided by the Penguin Karts team
SHA-256 (audited):
b2abfbf7f352dd3497d22685e8a20a80edc68a78d5182783504a5ef7608a638c

Within the scope of this audit Zokyo auditors have reviewed the following contract(s):

- Token.sol

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

- Implements and adheres to existing standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Follows best practices in efficient use of resources, 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 Penguin Karts 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. 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 or high issues found during the audit. Although certain number of low and informational issues were discovered, they relate to the following:

- Missed messages in exception$
- Constant can be used
- Centralization risk

After recommendations by Zokyo auditors and discussion with the Customer it was decided that all these issues don’t influence security and efficiency of the smart contract provided by Penguin Karts Team. By the Customer, ownable fuctionality will be used only after the listing, thus it is acceptable.

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 contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

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.

COMPLETE ANALYSIS

LOW | UNRESOLVED

Missed messages in exceptions

Token.sol. In case of discrepancy in requirements the error will not return a description of exception. It will make it harder to reveal a cause of revert.

Recommendation:
Use messages in “require” statements to add a description of exceptions.

INFORMATIONAL | UNRESOLVED

Constant can be used

Token.sol, Line 29. Token amount calculation can be moved to the public constant in order to save gas during the deployment.

Recommendation:
Consider using the public constant.

INFORMATIONAL | UNRESOLVED

Centralization risk

Token.sol. Contract uses onlyOwner modifier to control access to admin’s functionality. In case of losing access to the owner's address or sharing access with an unwanted person, admin can lose access to admin’s functionality.

Recommendation:
Consider using multisig as an owner’s address.

Post-audit:
By the client, ownable functionality will be used single time after the listing

	Token.sol
Re-entrancy	Pass
Access Management Hierarchy	Pass
Arithmetic Over/Under Flows	Pass
Delegatecall Unexpected Ether	Pass
Default Public Visibility	Pass
Hidden Malicious Code	Pass
Entropy Illusion (Lack of Randomness)	Pass
External Contract Referencing	Pass
Short Address/ Parameter Attack	Pass
Unchecked CALL Return Values	Pass
Race Conditions / Front Running	Pass
General Denial Of Service (DOS)	Pass
Uninitialized Storage Pointers	Pass
Floating Points and Precision	Pass
Tx.Origin Authentication	Pass
Signatures Replay	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Secured team

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

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

```
Contract: Token.sol
  Initialization->
    ✓ Mints proper amount (122ms)
  Methods->
    ✓ Transfers proper amount with antisnipe and liquidity restriction enabled (633ms)
    ✓ Disables liquidity restriction (289ms)
    ✓ Can't disable liquidity restriction if it's already disabled (201ms)
    ✓ Transfers proper amount with antisnipe enabled only (570ms)
    ✓ Transfers proper amount liquidity restriction enabled only (498ms)
    ✓ Disables antisnipe (222ms)
    ✓ Can't disable antisnipe if it's already disabled (194ms)
    ✓ Transfers proper amount without restrictions (435ms)
  9 passing (4s)
```

FILE	% STMTS	% BRANCH	% FUNCS
Token.sol	100.00	85.71	100.00
All files	**100**	**85.71**	**100**

We are grateful to have been given the opportunity to work with the Penguin Karts 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 Penguin Karts team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.