AUDIT REPORT

KoinWave - Jul 2024

Audit conducted by

RICARDO PONTES

Summary

Auditing Firm	CryptoHub
Architecture	CryptoHub Auditing Standard
Audit Conducted by	Ricardo | Blockchain Developer
Report Date	July 30, 2024

Audit Summary

CryptoHub team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

★ **KoinWave Token** smart contract source code has LOW RISK SEVERITY.
★ **KoinWave Token** has PASSED the smart contract audit.

For the detailed understanding of risk severity, source code vulnerability, and functional test, kindly refer to the audit.

✅Verify the authenticity of this report on CryptoHub Website: https://www.cryptohub.agency/

Table Of Contents

Project Overview

CryptoHub was consulted by KoinWave to conduct the smart contract security audit of their source code.

Project	KoinWave
Blockchain	BSC
Language	Solidity
CA	0x19db7d3c86ecbeccda7e1d0d42907d449b031291
Website:	https://koinwavetoken.com/

Public logo:

Solidity Source Code On Blockchain **(Verified Contract Source Code)**
https://bscscan.com/address/0x19db7d3c86ecbeccda7e1d0d42907d449b031291#code

Contract Name: ERC20Launch
Compiler Version: v0.8.19
Optimization Enabled: yes

Audit Methodology

The scope of this report is to audit the above smart contract source code and CryptoHub has scanned the contract and reviewed the project for common vulnerabilities, exploits, hacks, and back-doors. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

Smart Contract Vulnerabilities

- ☐ Re-entrancy
- ☐ Unhandled Exceptions
- ☐ Transaction Order Dependency
- ☐ Integer Overflow
- ☐ Unrestricted Action
- ☐ Incorrect Inheritance Order
- ☐ Typographical Errors
- ☐ Requirement Violation

Source Code Review

- ☐ Ownership Takeover
- ☐ Gas Limit and Loops
- ☐ Deployment Consistency
- ☐ Repository Consistency
- ☐ Data Consistency
- ☐ Token Supply Manipulation

Functional Assessment

- ☐ Access Control and Authorization
- ☐ Operations Trail and Event Generation
- ☐ Assets Manipulation
- ☐ Liquidity Access

Risk Classification

A contract is vulnerable if it has been flagged by a static analysis tool as such. As we will see later, this means that some contracts may be vulnerable because of a false-positive.

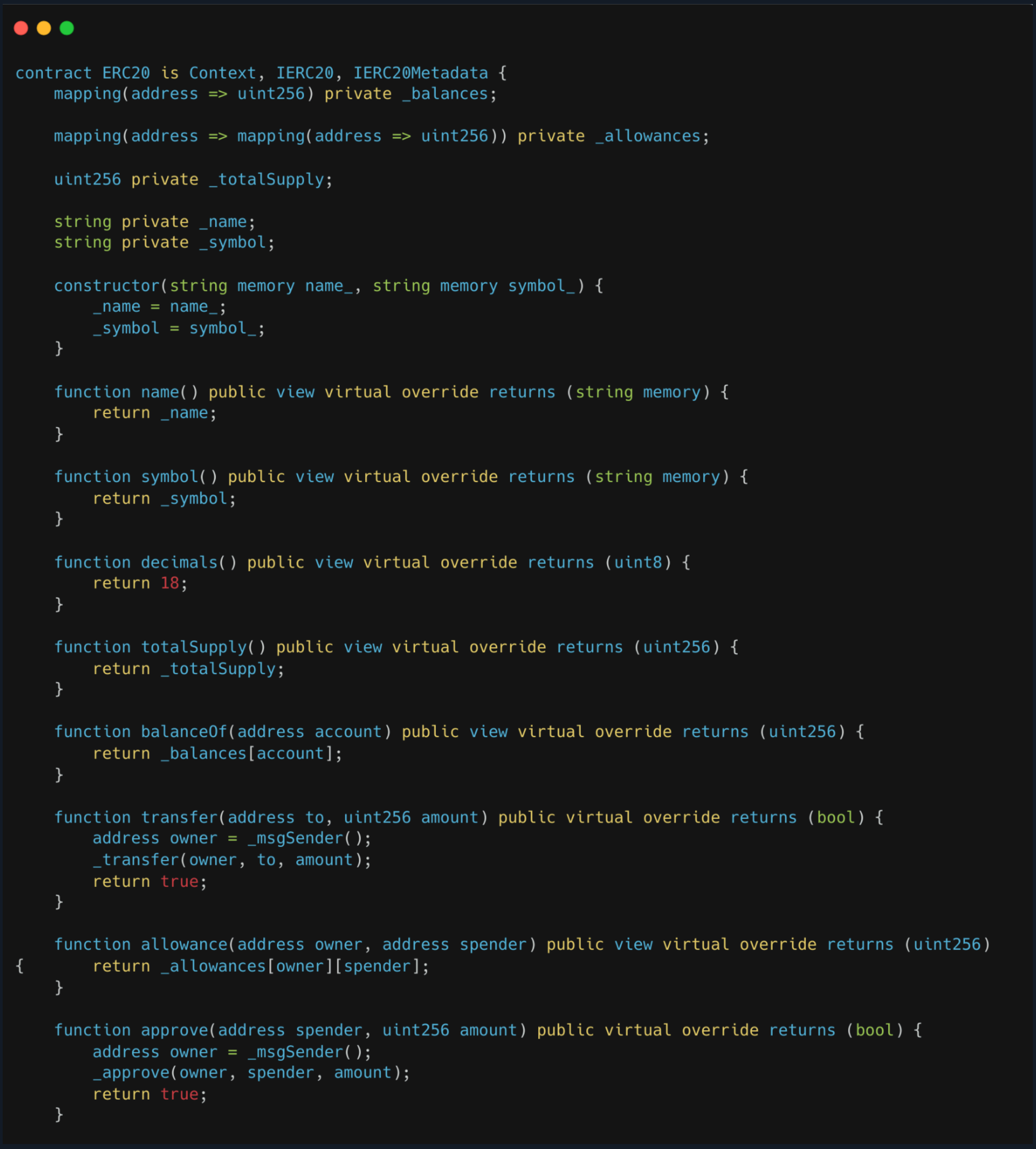
Exploitable: A contract is exploitable if it is vulnerable and the vulnerability could be exploited by an external attacker. For example, if the "vulnerability" flagged by a tool is in a function which requires to own the contract, it would be vulnerable but not exploitable.

Exploited: A contract is exploited if it received a transaction on the main network which triggered one of its vulnerabilities. Therefore, a contract can be vulnerable or even exploitable without having been exploited.

Risk severity	Meaning
! Critical	This level of vulnerability could be exploited easily, and can lead to asset loss, data loss, asset manipulation, or data manipulation. They should be fixed right away.
! High	These vulnerabilities are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to critical risk severity.
! Medium	This level of vulnerabilities should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution.
! Low	This level of vulnerability can be ignored. They are code style violations, and informational statements in the code. They may not affect the smart contract execution

Risk Assessment

Contract Snapshot



```
contract ERC20 is Context, IERC20, IERC20Metadata {
    mapping(address => uint256) private _balances;

    mapping(address => mapping(address => uint256)) private _allowances;

    uint256 private _totalSupply;

    string private _name;
    string private _symbol;

    constructor(string memory name_, string memory symbol_) {
        _name = name_;
        _symbol = symbol_;
    }

    function name() public view virtual override returns (string memory) {
        return _name;
    }

    function symbol() public view virtual override returns (string memory) {
        return _symbol;
    }

    function decimals() public view virtual override returns (uint8) {
        return 18;
    }

    function totalSupply() public view virtual override returns (uint256) {
        return _totalSupply;
    }

    function balanceOf(address account) public view virtual override returns (uint256) {
        return _balances[account];
    }

    function transfer(address to, uint256 amount) public virtual override returns (bool) {
        address owner = _msgSender();
        _transfer(owner, to, amount);
        return true;
    }

    function allowance(address owner, address spender) public view virtual override returns (uint256)
{       return _allowances[owner][spender];
    }

    function approve(address spender, uint256 amount) public virtual override returns (bool) {
        address owner = _msgSender();
        _approve(owner, spender, amount);
        return true;
    }
```

SWC Attacks

The following table contains an overview of the SWC registry. Each row consists of an SWC identifier (ID), weakness title, CWE parent and list of related code samples.

The auditor used a MythX tool, A static analyzer that parses the Soldity AST, a symbolic analyzer that detects possible vulnerable states, and a greybox fuzzer that detects vulnerable execution paths.

ID	Description	Status
SWC - 100	Function Default Visibility	✓ Passed
SWC - 101	Integer Overflow and Underflow	✓ Passed
SWC - 102	Outdated Compiler Version	✓ Passed
SWC - 103	Floating Pragma	✓ Passed
SWC - 104	Unchecked Call Return Value	✓ Passed
SWC - 105	Unprotected Ether Withdrawal	✓ Passed
SWC - 106	Unprotected SELFDESTRUCT Instruction	✓ Passed
SWC - 107	Reentrancy Passed	✓ Passed
SWC - 108	State Variable Default Visibility	✓ Passed
SWC - 109	Uninitialized Storage Pointer	✓ Passed
SWC - 110	Assert Violation Passed	✓ Passed
SWC - 111	Use of Deprecated Solidity Functions	✓ Passed
SWC - 112	Delegatecall to Untrusted Callee	✓ Passed
SWC - 113	DoS with Failed Call	✓ Passed
SWC - 114	Transaction Order Dependence	✓ Passed
SWC - 115	Authorization through tx.origin	✓ Passed
SWC - 116	Block values as a proxy for time	✓ Passed
SWC - 117	Signature Malleability	✓ Passed
SWC - 118	Incorrect Constructor Name	✓ Passed

SWC - 119	Shadowing State Variables	✓ Passed
SWC - 120	Weak Sources of Randomness from Chain	✓ Passed
SWC - 121	Protection against Signature Replay	✓ Passed
SWC - 122	Lack of Proper Signature Verification	✓ Passed
SWC - 123	Requirement Violation Passed	✓ Passed
SWC - 124	Write to Arbitrary Storage Location	✓ Passed
SWC - 125	Incorrect Inheritance Order Passed	✓ Passed
SWC - 126	Insufficient Gas Griefing	✓ Passed
SWC - 127	Arbitrary Jump with Function Type Variable	✓ Passed
SWC - 128	DoS With Block Gas Limit	✓ Passed
SWC - 129	Typographical Error	✓ Passed
SWC - 130	Right-To-Left-Override control character	✓ Passed
SWC - 131	Presence of unused variables	✓ Passed
SWC - 132	Unexpected Ether balance	✓ Passed
SWC - 133	Hash Collisions With Variable Arguments	✓ Passed
SWC - 134	Message call with hardcoded gas amount	✓ Passed
SWC - 135	Code With No Effects	✓ Passed
SWC - 136	Unencrypted Private Data On-Chain	✓ Passed

Our Findings

Low Severity Issues:

1 | Redundant checks

Type:	gas/code quality
Level:	Low
File:	contracts/libraries/LibEnsureSafeTransfer.sol

Functions:
safeTransferFromEnsureExactAmount, transferEnsureExactAmount, safeTransferFrom, safeTransfer

Description: all these functions use a check for address zero on the token,recipient and sender address which is redundant as the SafeERC20 library and the ERC20 already make both those checks.

Recommendation: Remove those checks.

Report Summary

Crypto Hub team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks.

★ **KoinWave Token** smart contract source code has LOW RISK SEVERITY.
★ **KoinWave Token** has PASSED the smart contract audit.

Risk severity	**Meaning**
! Critical	None critical severity issues identified
! High	None high severity issues identified
! Medium	None medium severity issues identified
! Low	1 low severity issue identified
Safety Score	96 out of 100

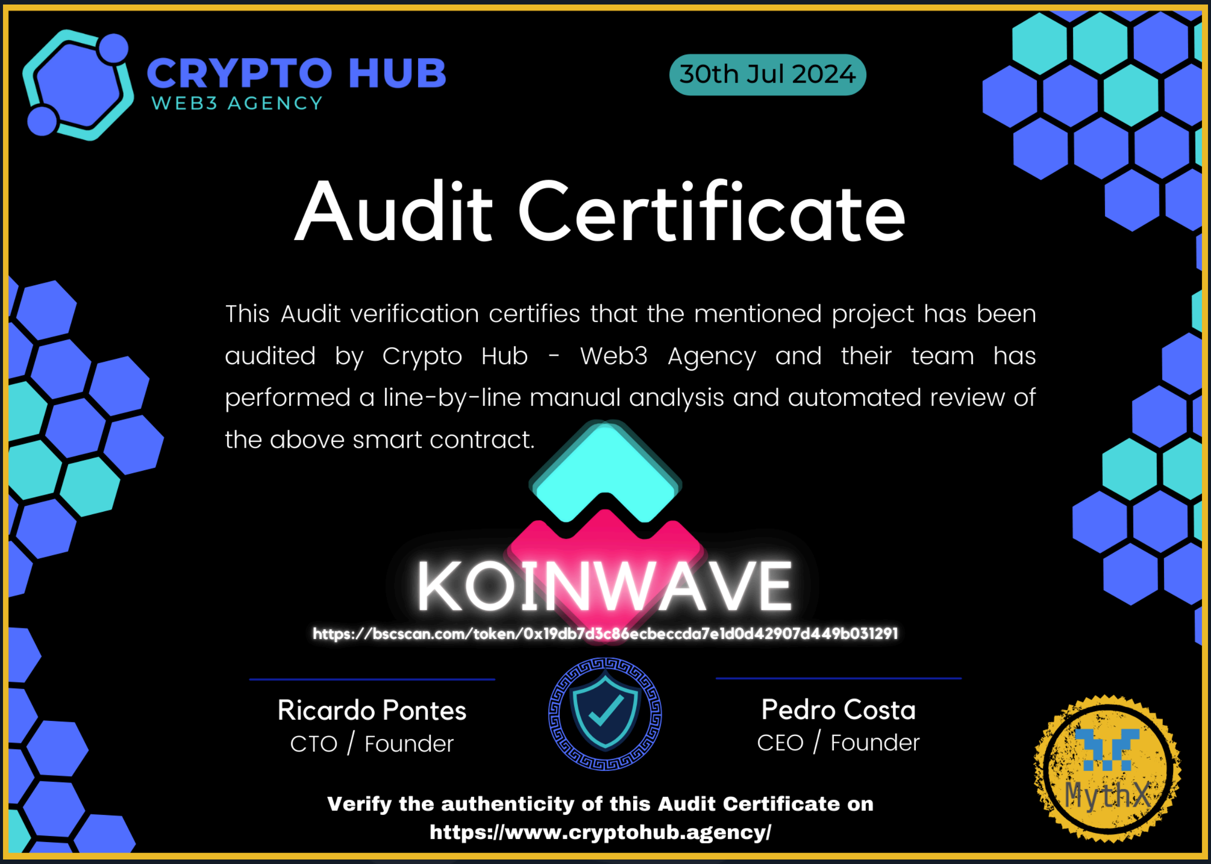
Note for stakeholders: Be aware that active smart contract owner privileges constitute an elevated impact on smart contract's safety and security. Make sure that the project team's KYC/identity is verified by an independent firm, e.g., CryptoHub. Always check if the contract's liquidity is locked.

A longer liquidity lock plays an important role in the project's longevity. It is recommended to have multiple liquidity providers. Examine the unlocked token supply in the owner, developer, or team's private wallets. Understand the project's tokenomics, and make sure the tokens outside of the LP Pair are vested or locked for a longer period of time.

Ensure that the project's official website is hosted on a trusted platform, and is using an active SSL certificate. The website's domain should be registered for a longer period of time.

Audit & KYC Certificates

We hereby certificate **KoinWave Token** smart contract as an audited project under the CryptoHub enterprise umbrella. And to represent it as such we issued the following certificate:



CRYPTO HUB
WEB3 AGENCY

30th Jul 2024

Audit Certificate

This Audit verification certifies that the mentioned project has been audited by Crypto Hub - Web3 Agency and their team has performed a line-by-line manual analysis and automated review of the above smart contract.

KOINWAVE

https://bscscan.com/token/0x19db7d3c86ecbeccda7e1d0d42907d449b031291

Ricardo Pontes
CTO / Founder

Pedro Costa
CEO / Founder

MythX

Verify the authenticity of this Audit Certificate on https://www.cryptohub.agency/

Legal Advisory

Important Disclaimer

CryptoHub provides contract auditing and project verification services for blockchain projects. The purpose of the audit is to analyze the on-chain smart contract source code, and to provide a basic overview of the project. This report should not be transmitted, disclosed, referred to, or relied upon by any person for any purposes without CryptoHub prior written consent.

CryptoHub provides the easy-to-understand assessment of the project, and the smart contract (otherwise known as the source code). The audit makes no statements or warranties on the security of the code. It also cannot be considered as an adequate assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have used all the data at our disposal to provide the transparent analysis, 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. Be aware that smart contracts deployed on a blockchain aren't resistant from external vulnerability, or a hack. Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's safety and security. Therefore, CryptoHub does not guarantee the explicit security of the audited smart contract.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

This report should not be considered as an endorsement or disapproval of any project or team. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. Do conduct your own due diligence and consult your financial advisor before making any investment decisions.

About CryptoHub

CryptoHub provides intelligent blockchain solutions. CryptoHub is developing an ecosystem that is seamless and responsive. Some of our services: Blockchain Security, Token Launchpad, NFT Marketplace, etc. CryptoHub's mission is to interconnect multiple services like Blockchain Security, DeFi, Gaming, and Marketplace under one ecosystem that is seamless, multi-chain compatible, scalable, secure, fast, responsive, and easy-to-use.

CryptoHub is built by a decentralized team of UI experts, contributors, engineers, and enthusiasts from all over the world. Our team currently consists of 3+ core team members, and 6+ casual contributors. CryptoHub provides manual, static, and automatic smart contract analysis, to ensure that the project is checked against known attacks and potential vulnerabilities.