MH

AUDITS

MH SWIFTSCAN REVIEW

KOI NETWORK

JUNE 28 TH 2022

TABLE OF CONTENTS

LEGAL DISCLAIMER

MH Audits are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts MH Audits to perform a security review.

MH Audits does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

MH Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

The report is provided only for the contract(s) mentioned in the report and does not include any other potential additions and/or contracts deployed by Owner. The report does not provide a review for contract(s), applications and/or operations, that are out of this report scope.

MH Audits' goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

MH Audits represents an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. MH Audits' position is that each company and individual are responsible for their own due diligence and continuous security.

The security audit is not meant to replace functional testing done before a software release. As one audit-based assessment cannot be considered comprehensive, we always recommend proceeding with several independent manual audits and a public bug bounty program to ensure the security of the smart contracts.

INTRODUCTION

MH Audits is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

Secure your project with MH Audits
We offer field-proven audits with in-depth reporting and a range of suggestions to improve and avoid contract vulnerabilities.

Industry-leading comprehensive and transparent smart contract auditing on all public and private blockchains.

Vunerability checking
A crucial manual inspection carried out to eliminate any code flaws and security loopholes. This is vital to avoid vulnerabilities and exposures incurring costly errors at a later stage.

Contract verification
A thorough and comprehensive review in order to verify the safety of a smart contract and ensure it is ready for launch and built to protect the end-user.

Risk assessment
Analyse the architecture of the blockchain system to evaluate, assess and eliminate probable security breaches. This includes a full assessment of risk and a list of expert suggestions.

In-depth reporting
A truly custom exhaustive report that is transparent and depicts details of any identified threats and vulnerabilities and classifies those by severity.

Fast turnaround
We know that your time is valuable and therefore provide you with the fastest turnaround times in the industry to ensure that both your project and community are at ease.

Best-of-class blockchain engineers
Our engineers combine both experience and knowledge stemming from a large pool of developers at our disposal. We work with some of the brightest minds that have audited countless smart contracts over the last 4 years.

PROJECT SUMMARY

PROJECT INTRODUCTION

Koi Network aims to build the culture bridge between Web2 and Web3 economy. It is a decentralized community-driven platform that enables users to merge Web2 applications and Web3 native cultures via a single touchpoint, bringing your digital collectibles to the masses.

Koi Network enables users to create, own, share and distribute their digital collectibles in a decentralized and disruptive way, including art, gaming and entertainment, etc. Identify, Transact & Scale with KOI.

Project Name *KOI Network*

Contract Name *KOI Token*

Contract Address *0xe84d9e32dC8cE819b8D6c83e50EDAfD46c6354dB*

Contract Chain *Mainnet*

Contract Type *Smart Contract*

Platform *EVM*

Language *Solidity*

Codebase *https://etherscan.com/address/0xe84d9e32dC8cE819b8D6c83e50EDAfD46c6354dB#code*

INFO & SOCIALS

Network *Ethereum (ERC20)*

Max Token Supply 1*.000.000.000*

Website *https://www.koi.io/*

Twitter *https://twitter.com/KoiMetaverse*

Telegram Chat *https://t.me/KoiMetaverse*

Telegram Ann *https://t.me/koi_announcement*

Discord *https://discord.gg/koi*

LinkedIn *https://www.linkedin.com/company/koi-metaverse/*

GitHub *https://github.com/KoiMetaverse*

Medium *https://koimetaverse.medium.com/*

EtherScan *https://etherscan.com/token/0xe84d9e32dC8cE819b8D6c83e50EDAfD46c6354dB*

AUDIT SCORES

85*

PASS

Issues	7
◆ Critical	0
◆ Major	0
◆ Medium	2
◆ Minor	4
◆ Informational	1
◆ Discussion	0

All issues are described in further detail on the following pages.

* Note that if no manual in-depth expert review has been performed a score multiplier of .9 will apply to the final result.

AUDIT SCOPE

FILE	LOCATION
contract.sol	*Ethereum Deployment: /address/0xe84d9e32dC8cE819b8D6c83e50EDAfD46c6354dB#code*

REVIEW METHODOLOGY

TECHNIQUES

This report has been prepared for KOI Network to discover issues and vulnerabilities in the source code of the KOI Network project as well as any contract dependencies that were not part of an officially recognized library. An examination has been performed, utilizing Static Analysis and MH SwiftScan review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts producedby industry leaders.

The security assessment resulted in findings that ranged from major to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective in the comments below.

TIMESTAMP

Version	v1.0
Date	2022/06/28
Description	Layout project
	Automated / Static security testing
	Summary

KEY FINDINGS

TITLE	SEVERITY	STATUS
Missing Exception On ERC20 Transfer Failure	◆ Medium	*Pending*
ERC20 Approve Front-Running Attack	◆ Medium	*Pending*
Use Of SafeMath Library	◆ Minor	*Pending*
Cheaper Inequalities In Require()	◆ Minor	*Pending*
Use Of Floating Pragma	◆ Minor	*Pending*
Cheaper Inequalities In If()	◆ Minor	*Pending*
Private Modifier Does Not Hide Data	◆ Informational	*Pending*

Description:

Functions of ERC-20 Token Standard should throw in special cases:

- transfer *should throw if the* _from *account balance does not have enough tokens to spend.*
- transferFrom *should throw unless the* _from *account has deliberately authorized the sender of the message via some mechanism.*

Location: *contract.sol L620-L629*

Issue: *Missing Exception On ERC20 Transfer Failure*

Level: *Medium*

Recommendation: *The ERC20 standard recommends throwing exceptions in functions* transfer *and* transferFrom*.*

SafeERC20 standard can also be used that automatically throws on failure.

Alleviation:

Description:

The approve() *method overrides current allowance regardless of whether the spender already used it or not, so there is no way to increase or decrease allowance by a certain value atomically unless the token owner is a smart contract, not an account.*

This can be abused by a token receiver when they try to withdraw certain tokens from the sender's account.

Meanwhile, if the sender decides to change the amount and sends another approve *transaction, the receiver can notice this transaction before it's mined and can extract tokens from both the transactions, therefore, ending up with tokens from both the transactions. This is a front-running attack affecting the* ERC20 Approve *function.*

Location: *contract.sol L51; L446-L449*

Issue: *ERC20 Approve Front-Running Attack*

Level: *Medium*

Recommendation: *Only use the approve function of the ERC-20 standard to change the allowed amount to 0 or from 0 (wait till transaction is mined and approved).*

*Token owner just needs to make sure that the first transaction actually changed allowance from N to 0, i.e., that the spender didn't manage to transfer some of N allowed tokens before the first transaction was mined. Such checking is possible using advanced blockchain explorers such as [Etherscan.io] (*https://etherscan.io/*)*

Another way to mitigate the threat is to approve token transfers only to smart contracts with verified source code that does not contain logic for performing attacks like described above, and to accounts owned by the people you may trust.

Alleviation:

IN-DEPTH VULNERABILITIES

Description:

SafeMath *library is found to be used in the contract. This increases gas consumption than traditional methods and validations if done manually.*

Also, Solidity 0.8 includes checked arithmetic operations by default, and this renders SafeMath *unnecessary.*

Location: *contract.sol L346*

Issue: *Use Of* SafeMath *Library*

Level: *Minor*

Recommendation: *We do not recommend using* SafeMath *library for all arithmetic operations. It is good practice to use explicit checks where it is really needed and to avoid extra checks where overflow/underflow is impossible.*

The compiler should be upgraded to Solidity version 0.8.0+ which automatically checks for overflows and underflows.

Alleviation:

Description:

The contract was found to be doing comparisons using inequalities inside the require statement. When inside the require statements, non-strict inequalities (>=, <=) are usually costlier than the strict equalities (>, <).

Location: *contract.sol L187; L202; L271*

Issue: *Cheaper Inequalities In* Require()

Level: *Minor*

Recommendation: *It is recommended to go through the code logic, and, if possible, modify the non-strict inequalities with the strict ones to save ~3 gas as long as the logic of the code is not affected.*

Alleviation:

IN-DEPTH VULNERABILITIES

Description:

Solidity source files indicate the versions of the compiler they can be compiled with using a pragma directive at the top of the solidity file. This can either be a floating pragma or a specific compiler version.

The contract was found to be using a floating pragma which is not considered safe as it can be compiled with all the versions described.

Location: *contract.sol L03*

Issue: *Use Of Floating Pragma*

Level: *Minor*

Recommendation: *It is recommended to follow the latter example, as future compiler versions may handle certain language constructions in a way the developer did not foresee. The developers should always use the exact Solidity compiler version when designing their contracts as it may break the changes in the future.*

pragma solidity ^0.4.17; not recommended -> compiles with 0.4.17 and above

pragma solidity 0.8.4; recommended -> compiles with 0.8.4 only

Alleviation:

Description:

The contract was found to be doing comparisons using inequalities inside the if statement. When inside the if statements, non-strict inequalities (>=, <=) are usually cheaper than the strict equalities (>, <).

Location: *contract.sol L126; L136*

Issue: *Cheaper Inequalities In* If()

Level: *Minor*

Recommendation: *It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save ~3 gas as long as the logic of the code is not affected.*

Alleviation:

IN-DEPTH VULNERABILITIES

Description:

Everything that is inside a contract is visible to all observers external to the blockchain. Making something `private` only prevents other contracts from reading or modifying the information, but it will still be visible to the whole world and observers of the blockchain.

Miners have access to all contracts' code and data. Developers must account for the lack of privacy in Ethereum.

Location: contract.sol L348-L356; L622

Issue: Private Modifier Does Not Hide Data

Level: Informational

Recommendation: Keep in mind that the `private` modifier does not make a variable invisible and should not keep sensitive contents within the modifier.

It is a best practice to use `private` when you really want to protect your state variables and functions because you hide them behind logic executed through internal or public functions.

Alleviation:

SOURCE CODE

https://etherscan.io/address/0xe84d9e32dC8cE819b8D6c83e50EDAfD46c6354dB#code

REPORT APPENDIX

FINDING CATEGORIES

The assessment process will utilize a mixture of static analysis, swift scan and other security techniques.

This report has been prepared for KOI Network project using MH SwiftScan to examine and discover vulnerabilities and safe coding practices in Supernova's smart contract including the libraries used by the contract that are not officially recognized.

The scan runs a comprehensive static analysis on the solidity code and finds vulnerabilities ranging from minor gas optimizations to major vulnerabilities leading to the loss of funds. The coverage scope pays attention to all the informational and critical vulnerabilities with over (110+) modules. The scanning and auditing process covers the following areas:

Various common and uncommon attack vectors will be investigated to ensure that the smart contracts are secure from malicious actors. The scanner modules find and flag issues related to gas optimizations that help in reducing the overall gas cost It scans and evaluates the codebase against industry best practices and standards to ensure compliance It makes sure that the officially recognized libraries used in the code are secure and up to date.

AUDIT SCORES

MH Audits AuditScores is not a live dynamic score. It is a fixed value determined at the time of the report issuance date.

*Note that if no manual in-depth expert review has been performed a score multiplier of .9 will apply to the final result.

MH Audits AuditScores are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports and scores are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts MH Audits to perform a security review.

AUDITS

WEBSITE
MHAUDITS.IO

TWITTER
@MHAUDITS