QuillAudits

Audit Report
September, 2022

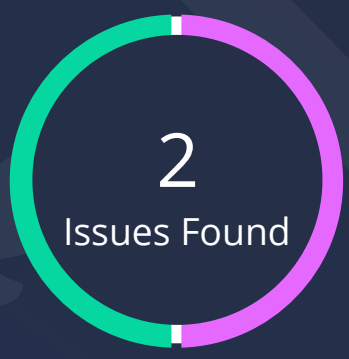
For

Vegas ONE

Table of Content

Executive Summary

Project Name	VegasONE
Overview	The contract is based on ERC721, but in order to save gas fee, they chose to inherit ERC721A. In addition it is an upgradeable contract and integrates the eip-712 signature scheme. Here poll creator creates poll calling createPoll() and voters can vote() for the poll creator. A poll can be in waiting, success, expired status. If a poll is successful then poll creator can mint NFT with the admin recommended signer.
Timeline	10 Aug, 2022 to 23 Aug,2022
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyze VegasONE OperatorNFT codebase for quality, security, and correctness. *https://github.com/taisys-technologies/audit-operator-nft/blob/master/contracts/operator/OperatorNFT.sol*
Fixed in	6cdf06b6647e235c147a9d1ae8244e60c6562932



High
Medium
Low
Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	1	1

Types of Severities

High

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level calls
- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Using block.timestamp
- Multiple Sends
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Testing

A. Contract - OperationNFT

High Severity Issues

No issues found

Medium Severity Issues

No issues found

Low Severity Issues

A1. Unnecessary import statement

Line	Function - transferAdmin

22-23	ftrace | funcSig
	`function transferAdmin(address newAdmin)`
	`external`
	`onlyRole(DEFAULT_ADMIN_ROLE)`
	`{`
	`if (newAdmin == address(0)) {`
	`revert AdminNotZero();`
	`}`
	`_grantRole(DEFAULT_ADMIN_ROLE, newAdmin);`
	`_revokeRole(DEFAULT_ADMIN_ROLE, _msgSender());`
	`emit TransferAdmin(newAdmin);`
	`}`
	`}`

Description

In the AccessControlUpgradableCustom, it was modified to help handle the management of ownership. In this function, admin roles can be transferred from one admin to the other. However, it comes with a risk when incorrect addresses are passed as the newAdmin and the administrative role is revoked automatically.

Remediation

A two factor mechanism could be adopted. transferAdmin to assign a role to a new address, while the original admin still has a role.
updateAdmin should be a function for acceptance of role and the former admin is revoked.

Status

Resolved

Informational Issues

A2. Absence of proper code commenting

Description

There were no comments provided for the contract. The devs or users who want to interact with code can misunderstand the intent behind a function hence the code needs proper commenting

Remediation

It is advised that the team provide proper comments for each and every function and variable, most preferably the natspec code commenting format.

Status

Resolved

Functional Testing

- ✓ Should get the contract address
- ✓ Should get the max token supply
- ✓ Should be able to change maxTokenSupply
- ✓ Should get the payment contract address
- ✓ Should be able to change signer address
- ✓ Should get the new signer of the operator NFT
- ✓ Should revert if not create poll is called when not in period (42ms)
- ✓ Should revert when create poll is paused
- ✓ Should revert when 0 is passed as the level Number
- ✓ Should revert if creator already created a poll before
- ✓ Should get poll created by a Raiser
- ✓ Should revert if period token supply is 0
- ✓ Should get poll created (105ms)
- ✓ Voter should not be able to vote to different poll creators in same period
- ✓ Should not be able to withdraw tokens if period is going on
- ✓ Should create poll and return correct values for getter functions
- ✓ Should not be able to change the maxTokenSupply when period started
- ✓ Should not be able to withdraw funds if period is over and not cancelled

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of the VegasONE. We performed our audit according to the procedure described above.

Some issues of Low severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture.In the End, VegasOne team resolved all issues.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the VegasONE Platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

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. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the VegasONE Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

600+
Audits Completed

$15B
Secured

600K
Lines of Code Audited

Follow Our Journey

Audit Report
September, 2022

For

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com