SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Audit

Security Assessment
14. December, 2021

For

VIKINGS CHAINS

Disclaimer

SolidProof.io reports 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. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc'...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof 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.

SolidProof.io Reports represent 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. SolidProof's position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	14. December 2021	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Binance Smart Chain (BEP20)

Website
https://vikingschain.com/

Telegram
https://t.me/VikingsChain

Twitter
https://twitter.com/VikingsChain

Instagram
https://www.instagram.com/vikingschain

Description

Legendary multiplayer battlegame that provides users to train heroes and win rewards in tournaments, utilizing the BSC technology.

Project Engagement

During the 7th of December 2021, **VikingsChain Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo

Contract Link

v1.0

- TBA

Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon aspossible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
    i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
    ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to SolidProof describe.

2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:

```
Address.sol
Context.sol
Counters.sol
ERC165.sol
ERC721.sol
ERC721Enumerable.sol
ERC721Holder.sol
IERC20.sol
IERC165.sol
IERC721.sol
IERC721Enumerable.sol
IERC721Metadata.sol
IERC721Receiver.sol
IVikingsChainNFT_Market.sol
IVikingsChainNFT.sol
Ownable.sol
Strings.sol
```

Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

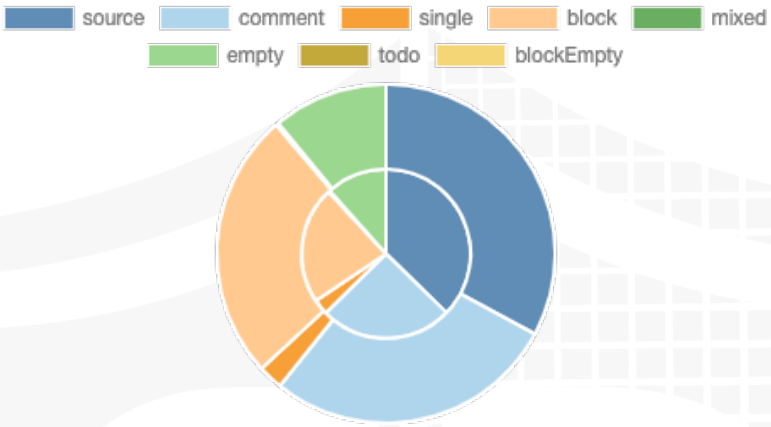
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

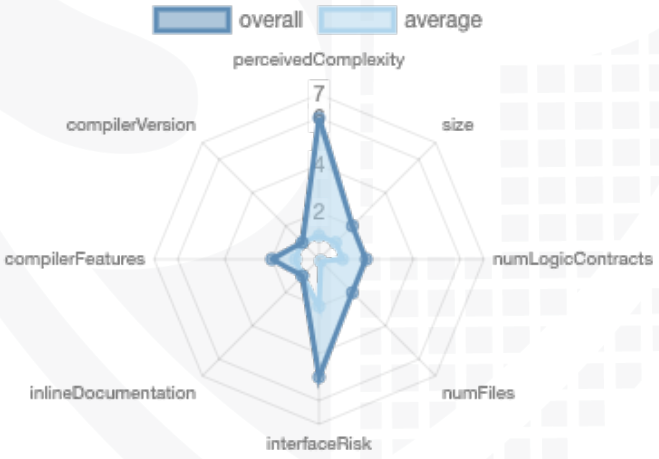
File Name	SHA-1 Hash
contracts/IERC721Enumerable.sol	81d5f0e8c3c8e57b863f0017c879bf3500c1d273
contracts/IVikingsChainNFT.sol	7354a55ffaf4d99e011cde3271780ce1acb42ec8
contracts/IERC20.sol	2a16c581fa3e71c55f5a07d494a1d6c4f937e6eb
contracts/IERC721Metadata.sol	7b4ee8ae808fc59bf00c2a51fefcba78ac3a631b
contracts/IERC721Receiver.sol	6de4d5d2874334a0bccba76b3be13aa438f5a99c
contracts/IERC165.sol	e27a07d950cc318a4221d878eb594485b93f165e
contracts/ERC165.sol	0dbde64740500a643d49e0a2298f63eff3d328b3
contracts/Strings.sol	6fa152156e90dc4cba54c8bade332cca3427a4f4
contracts/Context.sol	6cfff49179d5dd82ffa43390ff6ea2967ff6fa99
contracts/ERC721Enumerable.sol	e8e8210edae29a61f3a3f8f79139e449520adc5f
contracts/ERC721Holder.sol	ccdf3de7b13e892c16730f79c3cb873a0b9a829f
contracts/IVikingsChainNFT_Market.sol	1e8510491d343cf52f3930e383e6d8f1d4fdbb8b
contracts/IERC721.sol	0c664e8c5f9d4eaf4a0a8e22878076c573368592
contracts/ERC721.sol	1dee58f0a86a44a7722723d3fed84f644ed689a5
contracts/Address.sol	84a110f18c59ce35cfc78702e51c6340c22e9c6f
contracts/Ownable.sol	a7673da0172fe3d2f6facf2f48baeb14627b3d6d
contracts/VikingsChainNFT_Market.sol	31e992efc839477f8fd554dd10a6d3193acda955
contracts/Counters.sol	b72ed17629688a8bf4e4a4db437b500d09d08a60
contracts/VikingsChainNFT.sol	02d220c7f64b2e0ef5f6e6861ea018c7f47e0ec9

Metrics

Source Lines
v1.0



Risk Level
v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	4	3	8	4

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

Version	Public	Payable
1.0	79	4

Version	External	Internal	Private	Pure	View
1.0	39	104	6	4	47

State Variables

Version	Total	Public
1.0	26	8

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`^0.8.0`		`yes`	`yes` (3 asm blocks)	

Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/ Create/ Create2

1.0	`yes`		`yes`	`yes`		

Scope of Work

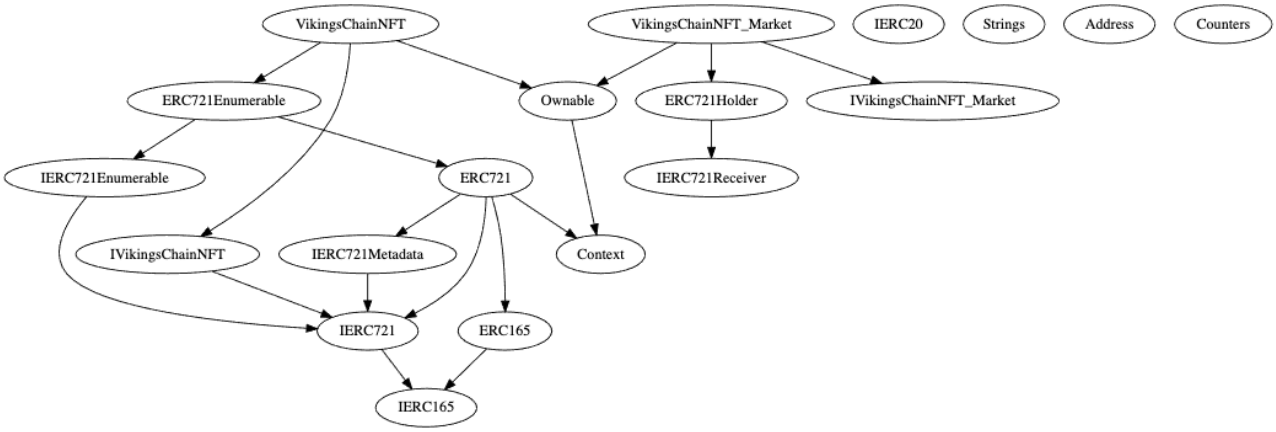
The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

1. Correct implementation of Token standard
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Overall checkup (Smart Contract Security)

Inheritance Graph

v1.0



Write functions of contract

VIKINGSCHAINNFT

- addCreator
- approve
- burnNFT
- mintAndApproveNFT
- mintNFT
- removeCreator
- renounceOwnership
- safeTransferFrom
- safeTransferFrom
- setApprovalForAll
- setBaseURI
- transferFrom
- transferOwnership

VIKINGSCHAINNFT_MARKET

- bid
- buyOnAuction
- listOnAuction
- onERC721Received
- renounceOwnership
- setFee
- setNextBidPercentage
- setTokenAddress
- transferOwnership
- withdraw

Modifiers

VikingsChainNFT_Market

- onlyOwner

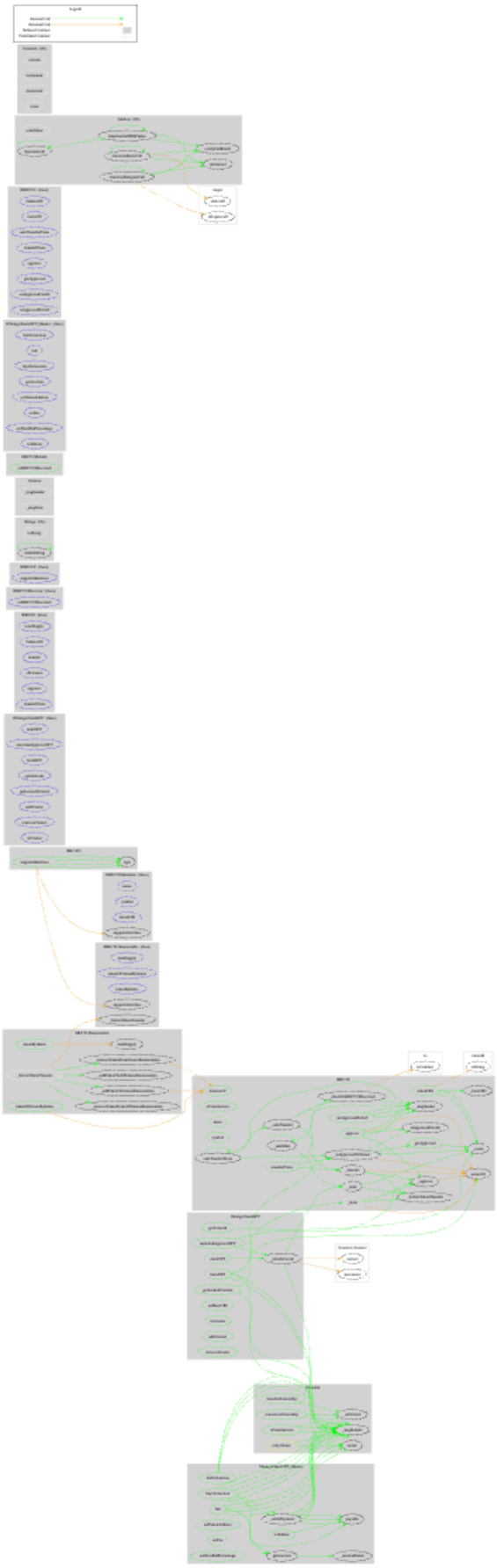
```
setTokenAddress
setFee
setNextBidPercentage
withdraw
```

VikingsChainNFT

- onlyOwner

```
setBaseURI
addCreator
removeCreator
```

CallGraph



Source Units in Scope
v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
🔍	contracts/IERC721Enumerable.sol	—	1	28	19	8	16	9	🔆
🔍	contracts/IVikingsChainNFT.sol	—	1	33	17	10	1	19	—
🔍	contracts/IERC20.sol	—	1	81	26	17	57	13	🔆
🔍	contracts/IERC721Metadata.sol	—	1	26	15	4	14	9	🔆
🔍	contracts/IERC721Receiver.sol	—	1	26	20	3	15	3	🔆
🔍	contracts/IERC165.sol	—	1	24	23	3	18	3	🔆
🎨	contracts/ERC165.sol	1	—	28	28	7	18	6	—
📚	contracts/Strings.sol	1	—	66	66	45	15	41	🔆
🎨	contracts/Context.sol	1	—	23	23	9	11	1	—
🎨	contracts/ERC721Enumerable.sol	1	—	162	158	65	73	39	—
📝	contracts/ERC721Holder.sol	1	—	27	22	7	12	5	🔆
🔍	contracts/IVikingsChainNFT_Market.sol	—	1	46	23	16	1	23	💰
🔍	contracts/IERC721.sol	—	1	142	62	40	99	21	🔆
📝	contracts/ERC721.sol	1	—	411	376	153	171	137	💻♻️
📚	contracts/Address.sol	1	—	216	171	80	113	47	💻👥
🎨	contracts/Ownable.sol	1	—	71	71	28	33	23	—
📝	contracts/VikingsChainNFT_Market.sol	1	—	189	189	133	1	136	💰🛋️
📚	contracts/Counters.sol	1	—	42	42	24	13	2	🔆Σ
📝	contracts/VikingsChainNFT.sol	1	—	108	108	83	2	77	🧮
📝📚🔍🎨	**Totals**	**11**	**8**	**1749**	**1459**	**735**	**683**	**614**	💻💰🛋️👥🧮🔆♻️Σ

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
nSLOC	normalized source lines of code (only source-code lines; no comments, no blank lines)
Comment Lines	lines containing single or block comments
Complexity Score	a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

Audit Results

AUDIT PASSED

Critical issues
- no critical issues found -

High issues
- no high issues found -

Medium issues
- no medium issues found -

Low issues
- no low issues found -

Issue	File	Type	Line	Description
#1	All	Contract doesn't import npm packages from source (like OpenZeppelin etc.)	-	We recommend to import all packages from npm directly without flatten the contract. Functions could be modified or can be susceptible to vulnerabilities
#2	All	A floating pragma is set	3	The current pragma Solidity directive is „"^0.8.0"".
#3	VikingsChainNFT_Market	Missing Zero Address Validation (missing-zero-check)	28, 129, 179	Check that the address is not zero
#4	VikingsChainNFT	State variable visibility is not set	14	It is best practice to set the visibility of state variables explicitly
#5	VikingsChainNFT_Market	Unchecked tokens transfer	158, 159, 160, 162, 163, 164, 73, 75	Use `SafeERC20`, or ensure that the transfer/transferFrom return value is checked

Informational issues
- no informational issues found -

Issue	File	Type	Line	Description
#1	Vikings ChainNFT_Market	Whitespaces	41, 43, 69,	The last argument must not be succeeded by any whitespace or comment. Remove the last whitespace
#2	Vikings ChainNFT_Market	Trailing whitespace	23, 52, 60, 79, 89, 103, 114, 151, 157, 174,	Line contains trailing whitespace. Remove whitespace
#3	Vikings ChainNFT_Market	Whitespace only on both sides	62	Assignment operator must have single space on both sides of it
#4	Vikings ChainNFT	Require statement needs error message	37, 38	Provide an error message for the require statement
#5	Vikings ChainNFT	Trailing whitespace	52	Line contains trailing whitespace. Remove whitespace
#6	Vikings ChainNFT	Whitespaces	82	The last argument must not be succeeded by any whitespace or comment. Remove the last whitespace

Audit Comments

14. December 2021:

.

SWC Attacks

ID	Title	Relationships	Status
SWC-136	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SWC-135	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SWC-133	Hash Collisions With Multiple Variable Length	CWE-294: Authentication Bypass by Capture-replay	PASSED
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SWC-127	Arbitrary Jump with Function Type	CWE-695: Use of Low-Level Functionality	PASSED
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SWC-120	Weak Sources of Randomness from Chain	CWE-330: Use of Insufficiently Random Values	PASSED
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	PASSED
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	NOT PASSED
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	NOT PASSED
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	NOT PASSED
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY