SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY

Kondux

Audit

Security Assessment
09. June, 2023

For

KONDUX

SolidProof_io @solidproof_io

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	28. May 2023	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	09. June 2023	• Reaudit

Note - This Audit report consists of a security analysis of the **KONDUX** smart contracts. This analysis did not include functional testing (or unit testing) of the contract’s logic.

Network

Ethereum

Website

kondux.info

Telegram

https://t.me/Kondux

Twitter

https://twitter.com/Kondux_KNDX

Discord

https://discord.gg/FGZvKnJ5xe

YouTube

https://www.youtube.com/channel/UCVHdS23n3vNYei7No_q6svA

Medium

https://kondux-web3-technologies.medium.com/

Instagram

https://www.instagram.com/kondux_kndx/

Description

Kondux is a Web3 virtual design lab for artists and brands. We generate digital assets layered with secured technologies creating entry points into the Metaverse and NFT markets for businesses and individuals. Our mission is to create custom-fit SaaS solutions for gaming, art, music, design and manufacturing applications integrating blockchain solutions into new business categories. By connecting people to advanced creative tools, we are designing a pipeline for innovation and unlimited growth within Web3 environments. Kondux is developing the first Web3 Content Management System for all to use as a gateway into the Metaverse.

Project Engagement

During the Date of 25 February 2023, **Kondux Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical 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

KONDUX

Contract Link

v1.0

- Provided as Files

v1.1

- https://github.com/Kondux/smart_contracts/tree/main/contracts
- Commit: cc67542
- **Deployed Contracts -**
- **Staking -** https://etherscan.io/address/0x07E6F2239d6FbE2CE00747fCFb8344ebBf973BcC#code
- **Helix -** https://etherscan.io/address/0x69a4A1CD8F2f2c3500F64634B9d69C642e9A5CA4#code
- **Treasury -** https://etherscan.io/address/0xaD2E62E90C63D5c2b905C3F709cC3045AecDAa1E

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.

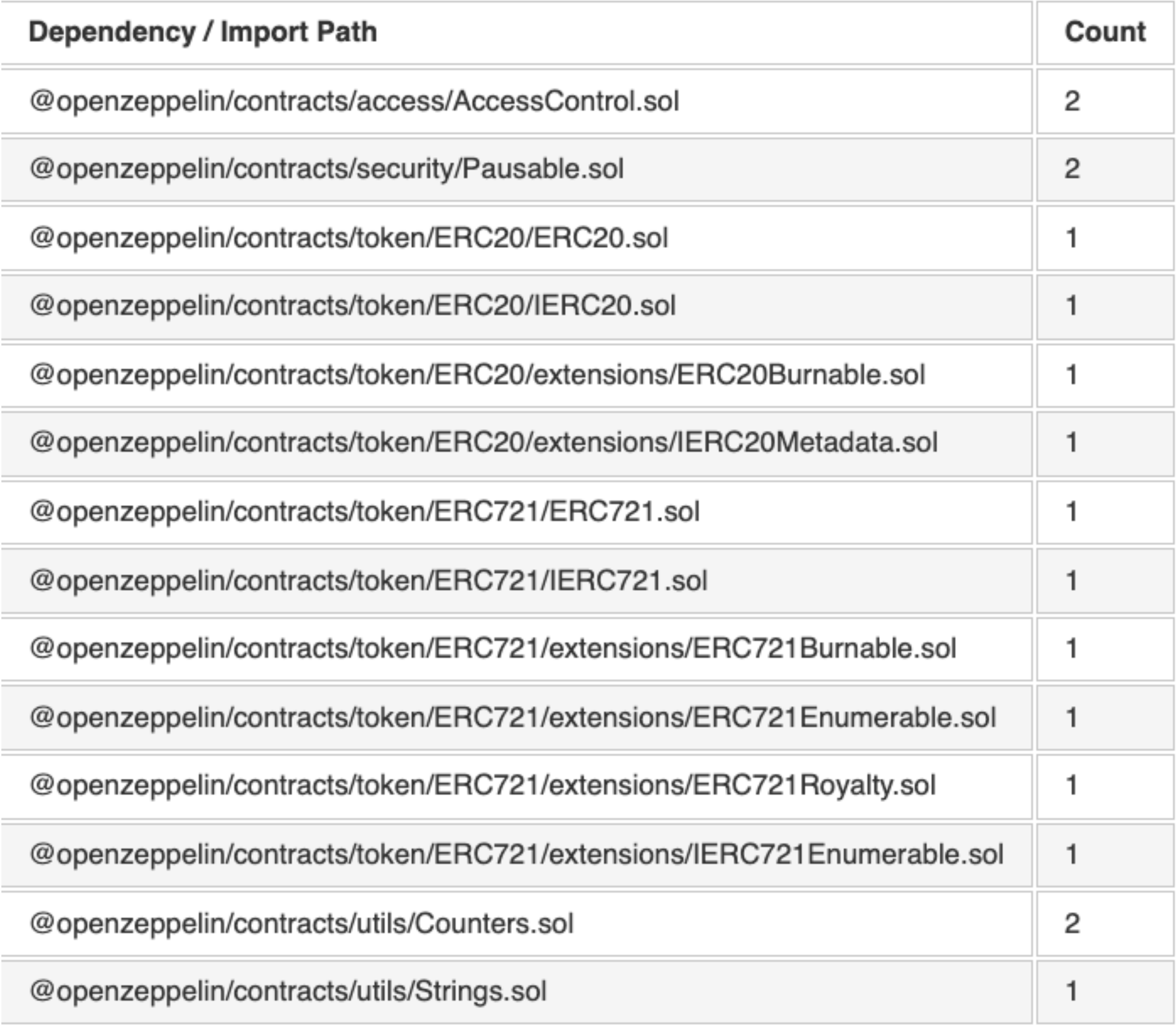
2. Testing and automated analysis that includes the following:
   i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii) Symbolic execution, which is analysing a program to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



Dependency / Import Path	Count
@openzeppelin/contracts/access/AccessControl.sol	2
@openzeppelin/contracts/security/Pausable.sol	2
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	1
@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol	1
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	1
@openzeppelin/contracts/token/ERC721/ERC721.sol	1
@openzeppelin/contracts/token/ERC721/IERC721.sol	1
@openzeppelin/contracts/token/ERC721/extensions/ERC721Burnable.sol	1
@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol	1
@openzeppelin/contracts/token/ERC721/extensions/ERC721Royalty.sol	1
@openzeppelin/contracts/token/ERC721/extensions/IERC721Enumerable.sol	1
@openzeppelin/contracts/utils/Counters.sol	2
@openzeppelin/contracts/utils/Strings.sol	1

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.

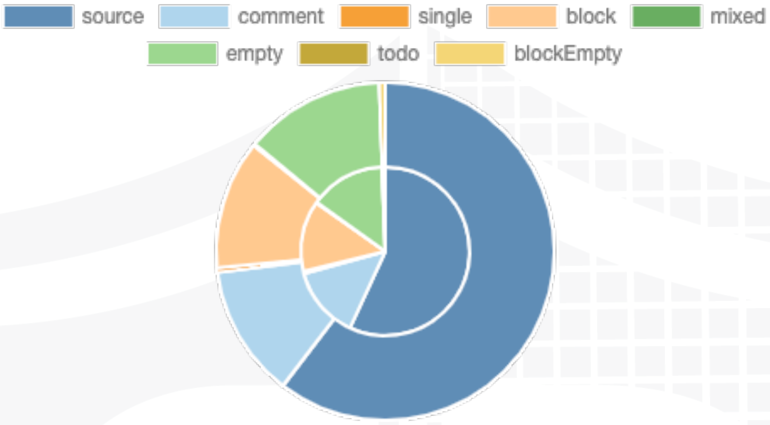
v1.0

File Name	SHA-1 Hash
contracts/Staking.sol	e9f597caf4b3937251af8402f790a36888592f47
contracts/Treasury.sol	35eff40bec4c2ce2e0a46249dc9beb328219427d
contracts/Authority.sol	6813f2028b140a7b63663959c564369233c7bcd3
contracts/Kondux_NFT.sol	c7b88fe0166e304f216f5c4cb9f22b0aa9b2757e
contracts/Helix.sol	3f75a57bcce0d00609efc74ba8647e00ddf68e02
contracts/types/AccessControlled.sol	387e40edca753894ebc167cbc20d8e37476d1609

Metrics

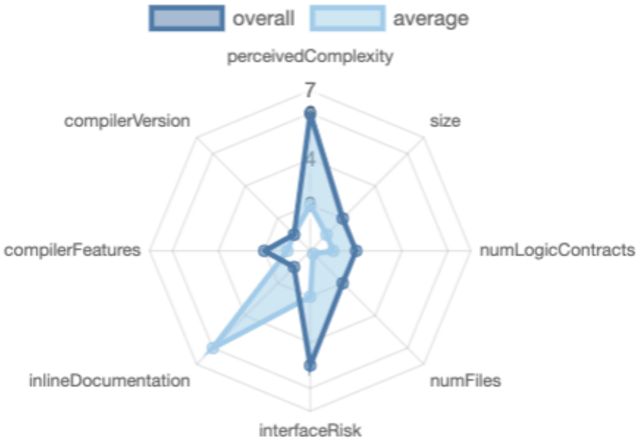
Source Lines

v1.0



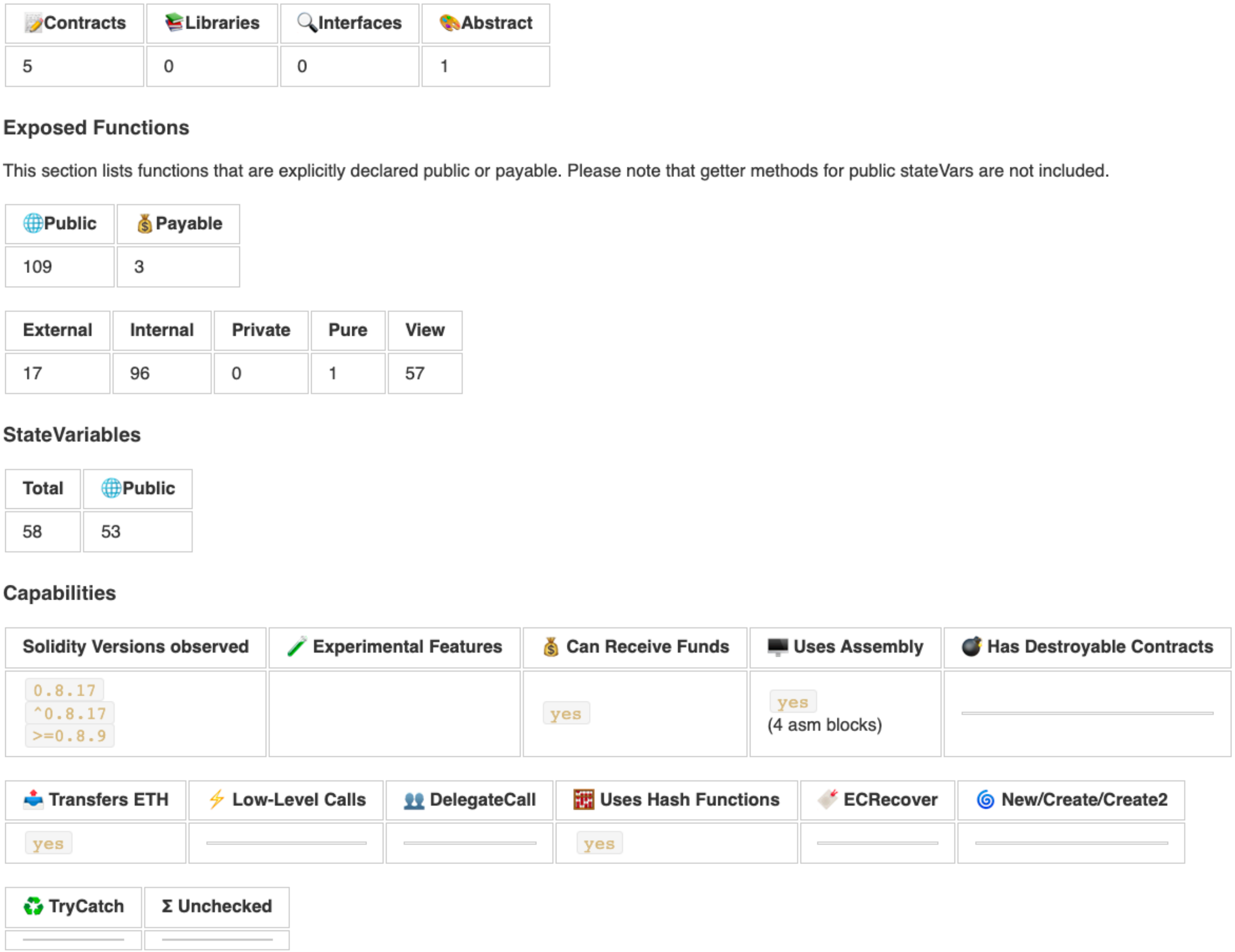
Risk Level

v1.0



Capabilities

Components



📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
5	0	0	1

Exposed Functions

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

🌐Public	💰Payable
109	3

External	Internal	Private	Pure	View
17	96	0	1	57

StateVariables

Total	🌐Public
58	53

Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts

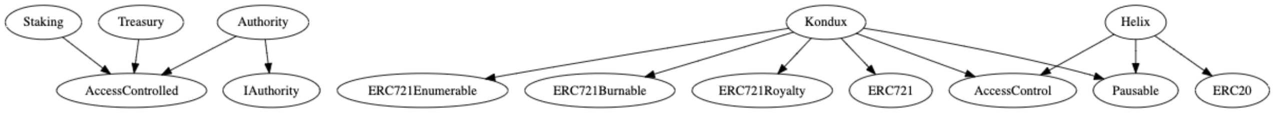
`0.8.17`
`^0.8.17`
`>=0.8.9`
		`yes`	`yes`
			(4 asm blocks)	

📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	🔖 ECRecover	🌀 New/Create/Create2
`yes`			`yes`		

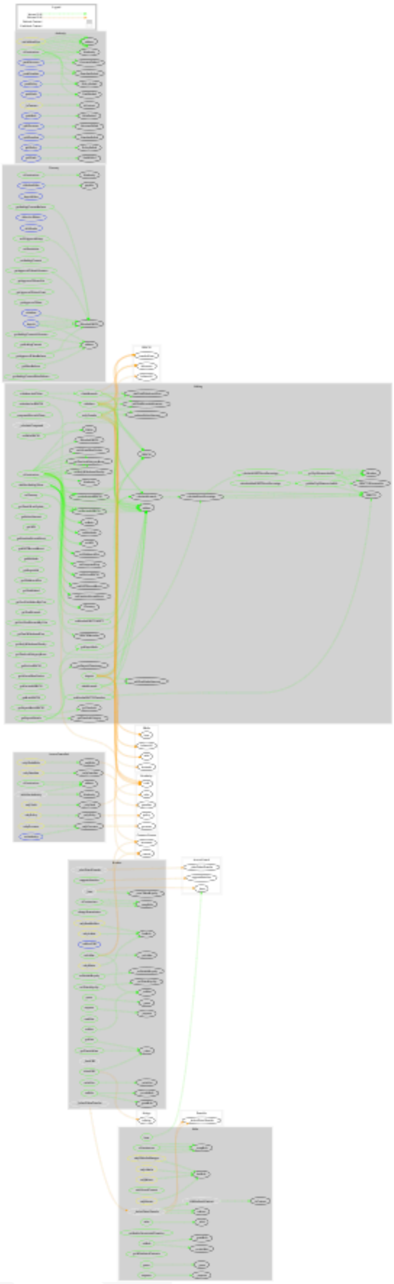
♻️ TryCatch	Σ Unchecked

Inheritance Graph

v1.0



CallGraph
v1.0



Scope of Work/Verify Claims

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. Is contract an upgradeable
2. Correct implementation of Token standard
3. Deployer cannot mint any new tokens
4. Deployer cannot burn or lock user funds
5. Deployer cannot pause the contract
6. Deployer cannot set fees
7. Deployer cannot blacklist/antisnipe addresses
8. Overall checkup (Smart Contract Security)

Is contract an upgradeable

Name	
Is contract an upgradeable?	No

Correct implementation of Token standard

ERC20				
Function	**Description**	**Exist**	**Tested**	**Verified**
TotalSupply	Provides information about the total token supply	✓	✓	✓
BalanceOf	Provides account balance of the owner's account	✓	✓	✓
Transfer	Executes transfers of a specified number of tokens to a specified address	✓	✓	✓
TransferFrom	Executes transfers of a specified number of tokens from a specified address	✓	✓	✓
Approve	Allow a spender to withdraw a set number of tokens from a specified account	✓	✓	✓
Allowance	Returns a set number of tokens from a spender to the owner	✓	✓	✓

ERC721				
Function	**Description**	**Exist**	**Tested**	**Verified**
BalanceOf	Count all NFTs assigned to an owner	✓	✓	✓
OwnerOf	Find the owner of an NFT	✓	✓	✓
SafeTransferFrom	Transfers the ownership of an NFT from one address to another address	✓	✓	✓
SafeTransferFrom	See above - Difference is that this function has an extra data parameter	✓	✓	✓
TransferFrom	Transfer ownership of an NFT	✓	✓	✓
Approve	Change or reaffirm the approved address for an NFT	✓	✓	✓
SetApprovalForAll	Enable or disable approval for a third party ("operator") to manage all of `msg.sender`'s assets	✓	✓	✓
GetApproved	Get the approved address for a single NFT	✓	✓	✓
IsApprovedForAll	Query if an address is an authorized operator for another address	✓	✓	✓
SupportsInterface	Query if a contract implements an interface	✓	✓	✓
Name	Provides information about the name	✓	✓	✓
Symbol	Provides information about the symbol	✓	✓	✓
TokenURI	Provides information about the TokenUri	✓	✓	✓

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✗
Max / Total Supply	N/A		

Comments:

v1.1

- MinterRole wallets/addresses can mint tokens to any arbitrary amount

Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock	–	–	–
Deployer cannot burn	✓	✓	✓

Comments:

v1.1

- Users can burn their NFTs and it will result in resetting of the royalty as well.

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer can pause	✓	✓	✗

Comments:

v1.1

- The admin address can Pause/Unpause transactions

Deployer cannot set fees

Name	Exist	Tested	Status
Deployer cannot set fees over 25%	✓	✓	✗
Deployer cannot set fees to nearly 100% or to 100%	✓	✓	✗

Comments:

v1.1

- Withdrawal Staking Fees can be set up to 100%

Deployer can blacklist/antisnipe addresses

Name	Exist	Tested	Status
Deployer can blacklist/antisnipe addresses	✓	✓	✗

Comments:

v1.1

- The owner is able to whitelist addresses, and only those addresses will be allowed to transfer Helix Tokens. On the other hand, the owner can toggle between enabling and disabling the Unrestricted Transfer flag. However, if it is enabled only then all the users will be able to make transfers.
- Moreover, this functionality is intended for whitelisting contracts but there is no such check to verify that the whitelisted address is a contract or not.

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

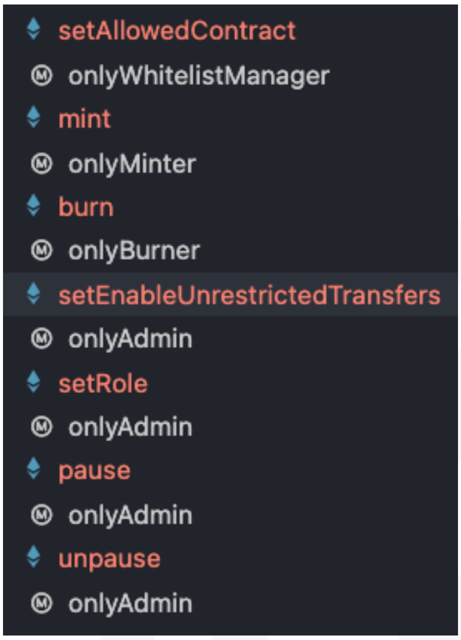
Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	🚩
Unverified / Not checked	✗
Not available	–

Modifiers and public functions

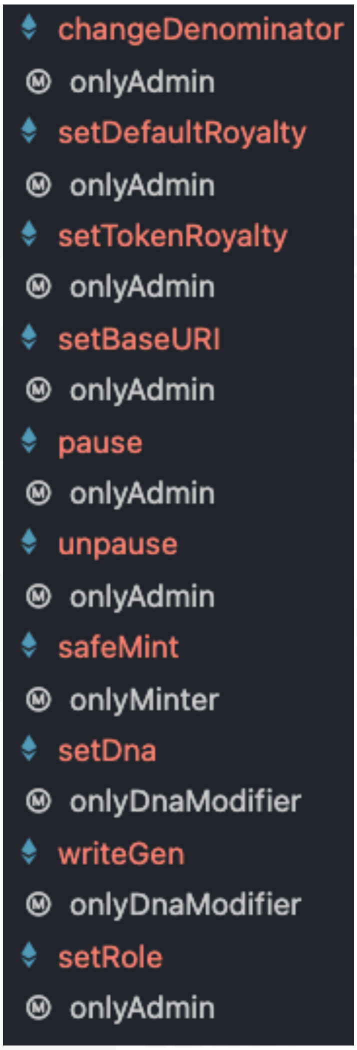
v1.0

Helix



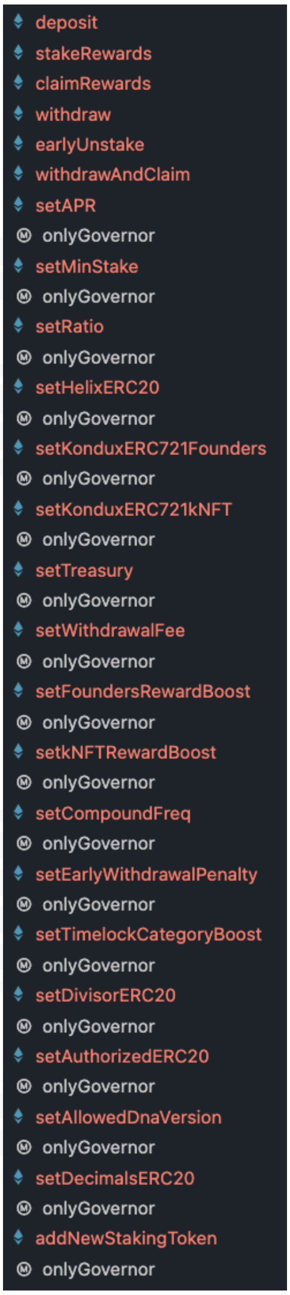
```
setAllowedContract
onlyWhitelistManager
mint
onlyMinter
burn
onlyBurner
setEnableUnrestrictedTransfers
onlyAdmin
setRole
onlyAdmin
pause
onlyAdmin
unpause
onlyAdmin
```

Kondux_NFT



```
changeDenominator
onlyAdmin
setDefaultRoyalty
onlyAdmin
setTokenRoyalty
onlyAdmin
setBaseURI
onlyAdmin
pause
onlyAdmin
unpause
onlyAdmin
safeMint
onlyMinter
setDna
onlyDnaModifier
writeGen
onlyDnaModifier
setRole
onlyAdmin
```

Staking



```
deposit
stakeRewards
claimRewards
withdraw
earlyUnstake
withdrawAndClaim
setAPR
onlyGovernor
setMinStake
onlyGovernor
setRatio
onlyGovernor
setHelixERC20
onlyGovernor
setKonduxERC721Founders
onlyGovernor
setKonduxERC721kNFT
onlyGovernor
setTreasury
onlyGovernor
setWithdrawalFee
onlyGovernor
setFoundersRewardBoost
onlyGovernor
setkNFTRewardBoost
onlyGovernor
setCompoundFreq
onlyGovernor
setEarlyWithdrawalPenalty
onlyGovernor
setTimelockCategoryBoost
onlyGovernor
setDivisorERC20
onlyGovernor
setAuthorizedERC20
onlyGovernor
setAllowedDnaVersion
onlyGovernor
setDecimalsERC20
onlyGovernor
addNewStakingToken
onlyGovernor
```

Authorised Address Privileges:

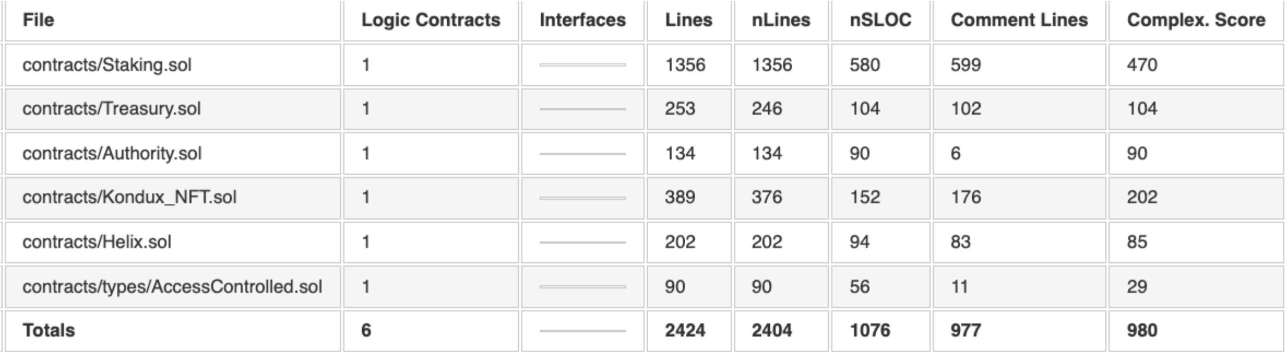
The address/wallets with the “onlyGovernor” “MINTER_ROLE”, “BURNER_ROLE”, and “ADMIN_ROLE” has the following privileges:

S.No	File	Privileges
#1	Helix.sol	• Wallets with the minter role can mint and burn tokens without any limitations. • Admin can grant these roles to any arbitrary addresses.
#2	Staking.sol (onlyGovernor)	• Set APR, Ratio and minimum stake amount for a given token ID to any arbitrary value includiong zero. • Set HelixERC20, and KonduxERC721 Founders, and kNFT addresses • Set treadury address • Set Withdrawal fee upto 100% • Set founders reward and kNFT reward boost to any arbitrary value • Set compound Frequency to any arbitrary value • Set early withdraw penalty upto 100% which is not recommended • Set divisor for a given ERC20 address • Authorize an ERC20 token for staking • Set decimals for an ERC20 • Set allowed DNA versions for staking
#3	Kondux_NFT.sol (onlyAdmin, and MINTER_ROLE)	• Set token royalty for a tokenID to any arbitrary address which means that it is possible to set/change the royalty receiving address of an NFT regardless of the fact that the recipient address has any connection with the NFT or not. • Set base URI • Wallets/Addresses with the Minter role mint NFTs • Set DNA value for a respective tokenID • Write range of Bytes to the DNA value of a given Token ID

S.No	File	Privileges
#4	Treasury.sol	• The governor address/wallet can grant permissions to accounts that they can deposit and withdraw tokens/ETH from the treasury contract. • Set staking contract address • Approve the treasury contract for the KonduxERC20 token transfers

Source Units in Scope

v1.0



File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/Staking.sol	1		1356	1356	580	599	470
contracts/Treasury.sol	1		253	246	104	102	104
contracts/Authority.sol	1		134	134	90	6	90
contracts/Kondux_NFT.sol	1		389	376	152	176	202
contracts/Helix.sol	1		202	202	94	83	85
contracts/types/AccessControlled.sol	1		90	90	56	11	29
Totals	**6**		**2424**	**2404**	**1076**	**977**	**980**

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalised lines of the source unit (e.g. normalises functions spanning multiple lines)
nSLOC	normalised 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

Critical issues

No critical issues

High issues

No high issues

Medium issues

Medium Issues Acknowledged

Issue	File	Type	Line	Description	Status
#1	Kondux_Founders.sol	Authority is able to set royalty address and value	103	The wallets with the “onlyAdmin” role can change the royalty amount of an NFT to any value including zero or a 100%. This is not recommended because there is no check to verify if the royalty address has any connection to the tokenID or not. We recommend to verify the connection between the address and the token ID according to the business logic.	**ACK**
#2	Staking.sol	Fees can be 100%	791	The “onlyGovernor" address is able to set the withdrawal fees up to a 100% which will cause the loss of user funds, and is not recommended.	**ACK**
#3	Helix.sol	Authority can Mint and Burn	132, 141	The addresses with minter and burner role can mint unlimited tokens at anytime, and burn tokens from any arbitrary address that holds the Helix token. This is not recommended as it may result in the loss of users’ funds	**ACK**
#4	Helix.sol	Owner can disable transfer	149	The admin can enable/disable transfers by toggling this function and all the addresses that are not in the whitelist won’t be able to make any transfers.	**ACK**

#5	Kondux_NFT.sol	Owner can disable transfer	146, 153	The admin can enable/disable transfers by toggling this function and all the addresses that are not in the whitelist won’t be able to make any transfers.	**ACK**

Low issues

Issue	File	Type	Line	Description	Status
#1	Staking.sol	Missing Zero Address Validation (missing-zero-check)	All setter functions with “address” parameter	Check that the address is not zero	**Fixed**
#2	Treasury.sol	Missing Zero Address Validation (missing-zero-check)	132, 151	Check that the address is not zero	**Fixed**

Informational issues

No informational issues

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information https://docs.soliditylang.org/en/latest/natspec-format.html) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variables, functions etc. do.

09. June 2023:

- There is still an owner (The owner still has not renounced ownership)
- We recommend **KONDUX** team conduct unit and fuzz tests thoroughly to rule out the possibilities of unwanted logical and calculation errors.
- The KONDUX Team has acknowledged the medium issues as they are part of their business logic.
- Read the whole report and modifiers section for more information

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 Arguments	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 Variable	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 Attributes	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	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	**PASSED**
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	**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**

SolidProof_io

@solidproof_io

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY