SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

v1.0: 17. January, 2022
v1.1: 20. January, 2022

Audit

Security Assessment

08. February, 2022

For

UNIVERSE

LAUNCH PLATFORM

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	17. January 2022	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	20. January 2022	• Reaudit
1.2	08. February 2022	• Reaudit

Network
Binance Smart Chain (BEP20)

Website
https://universeun.com/

Telegram
https://t.me/universeun

Twitter
https://twitter.com/daouniverse

Github
https://universeun.com/

Description

The cryptocurrency combined with NFT, Swap and IDO platforms is a more friendly NFT token. It's responsible for bringing the crypto world to more people, and it has a higher mission.

universeNFT is a card with the theme of ft planet, which not only has a real planet design, but also a virtual currency planet, fantasy planet and other planet design.

UniverseSWAP is a built-in swap platform, not only of which the exchange speed is extremely fast, but also who has the functions of pledge nft farm, mining, etc., and will continue to develop more functions in the future

U-Ido platform is a professional launch platform. We will help potential virtual currency to launch, and provide nft market and swap platform help

Project Engagement

During the 13th of January 2022, **UniverseUN 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

- Provided as files

v1.1 & v1.2

- Provided as files

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.

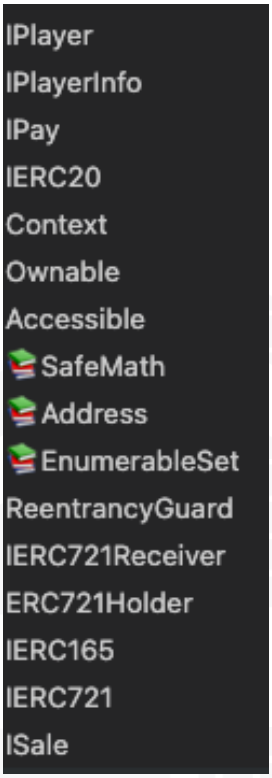
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

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

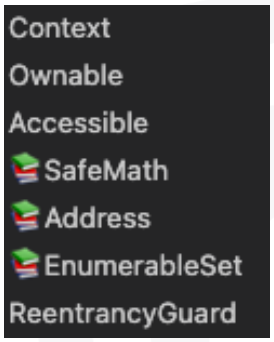
Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:

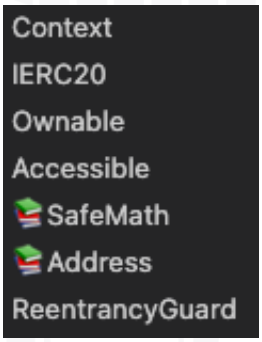
BoxSale



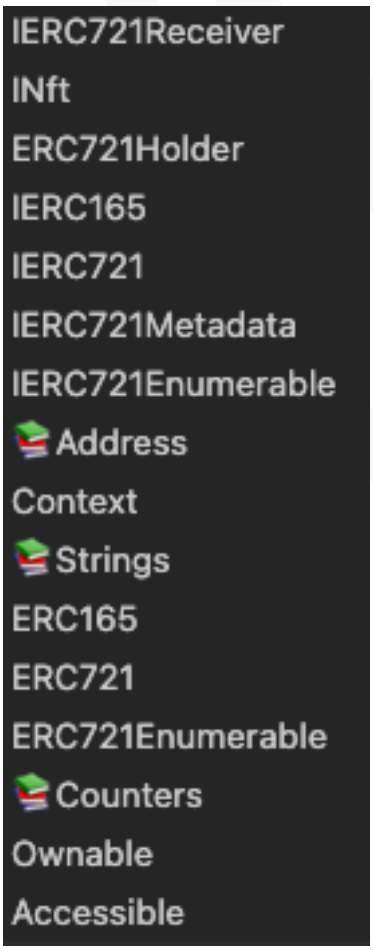
Box



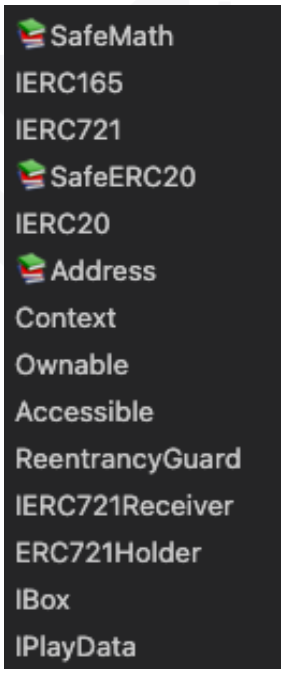
IdoAction



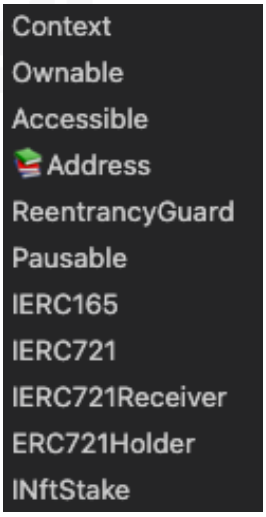
Nft



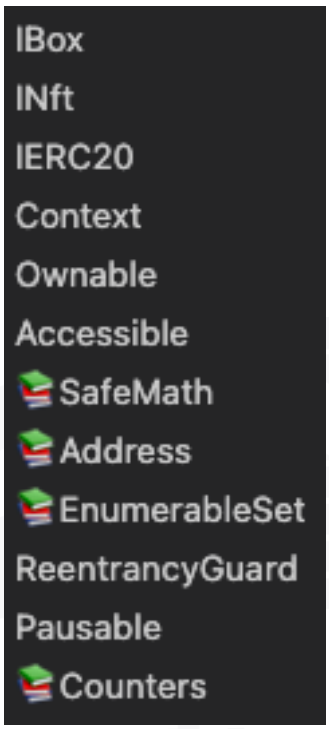
NFTStake



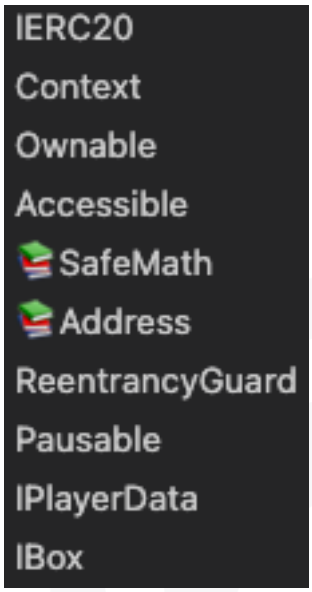
NFTStakeAction



PlayerDatas



PlayerAction



Tested Contract Files

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

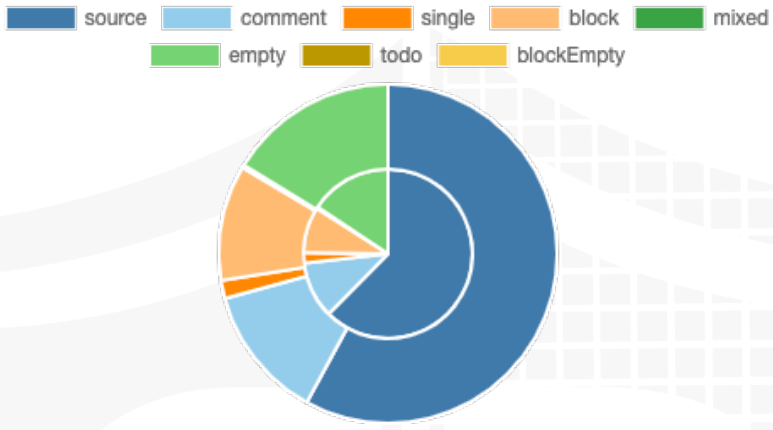
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

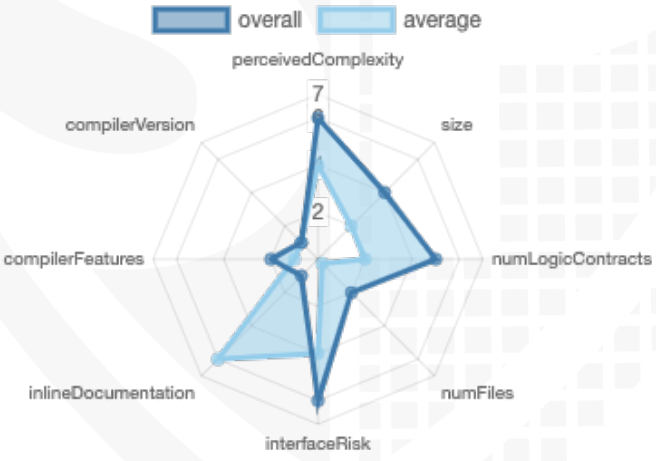
File Name	SHA-1 Hash
contracts/nftstake.sol	4d59ce000f06a70a684cd736ec0ec3357326b3b0
contracts/nftstakeAction.sol	dd36b7f2e6a217d0d58fbc98cf6acc0829492180
contracts/box.sol	363b42391d288e7ae60cc38ead989ee7affc3ce0
contracts/playerAction.sol	345dc07638b055d8316439ca9eb91dbae9b9d135
contracts/ldoAction.sol	ef9bed2cf32edf3a53718cd79c86bb9bcef88433
contracts/nft.sol	dd8f010ce2e058db533e0366717a93100eb572b4
contracts/boxSale.sol	1bbd305b99e40b67d01dddfeb54e0cf7ac42220d
contracts/playerDatas.sol	3f6fa5d775f27f83d85b3339cfbc44e6070f8a12

Metrics

Source Lines
v1.0



Risk Level
v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	20	21	31	29

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	269	1

Version	External	Internal	Private	Pure	View
1.0	118	553	34	84	193

State Variables

Version	Total	Public
1.0	116	46

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`^0.8.0`	`ABIEncoderV2`	`yes`	`yes` (17 asm blocks)	

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

1.0			`yes`	`yes`		

Scope of Work

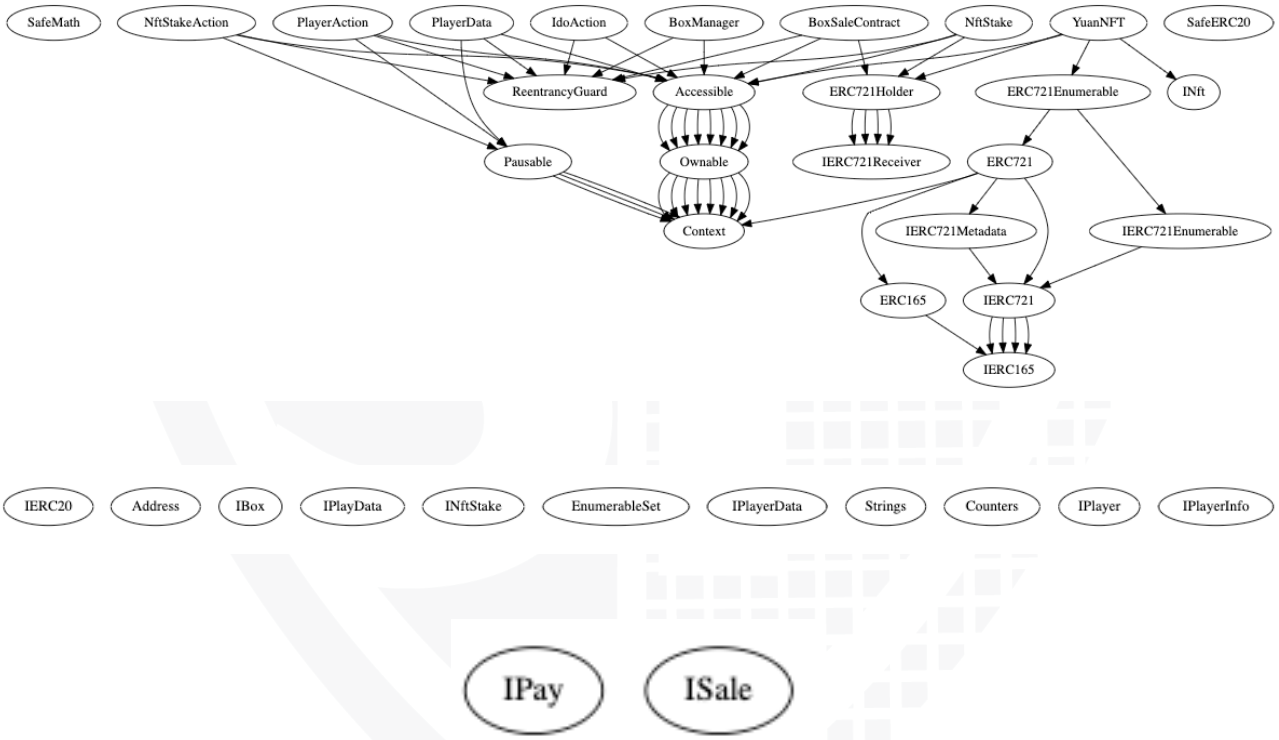
The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

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



Verify Claims

Correct implementation of Token standard

Tested	Verified
✓	✓

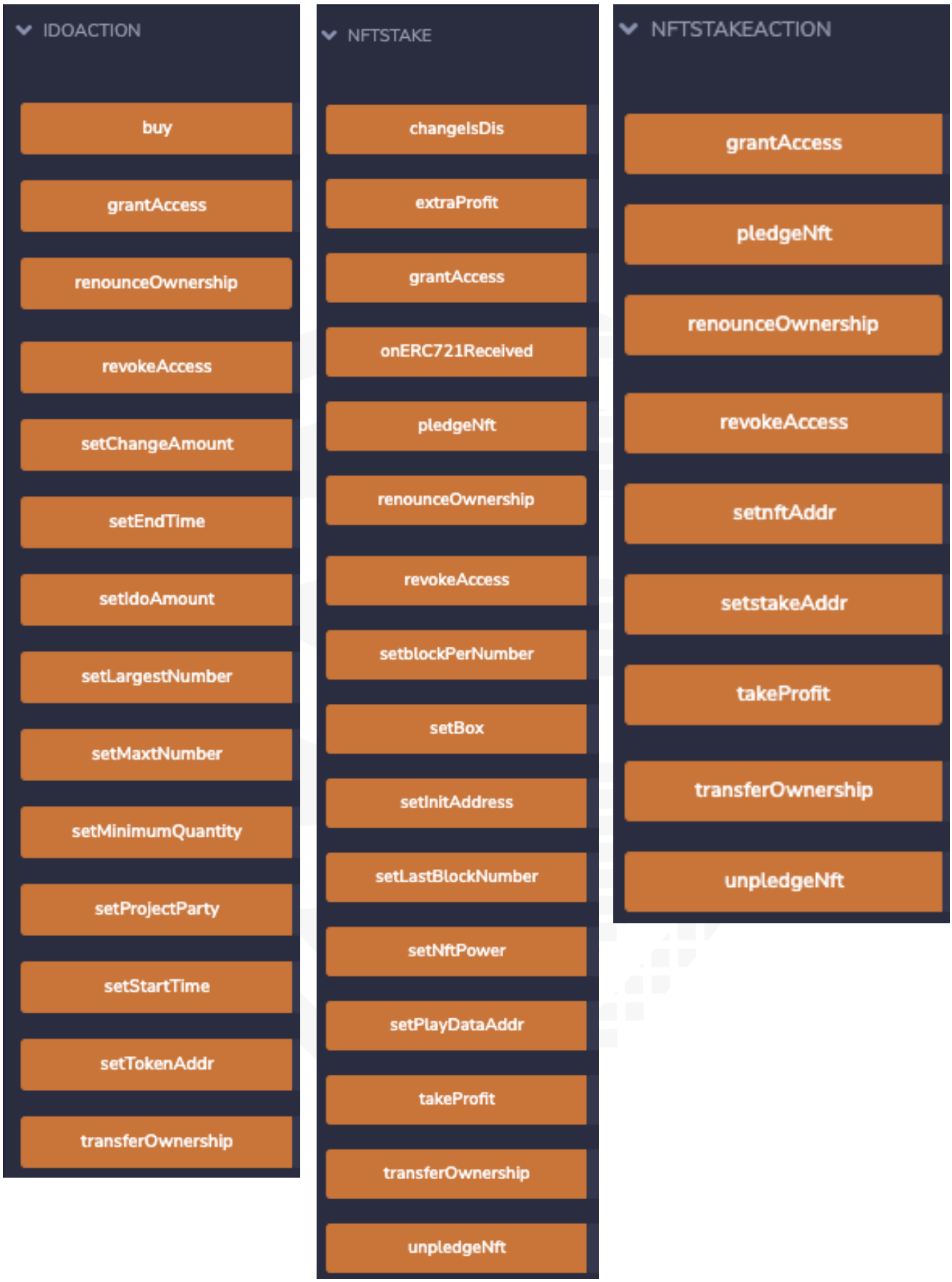
YuanNFT

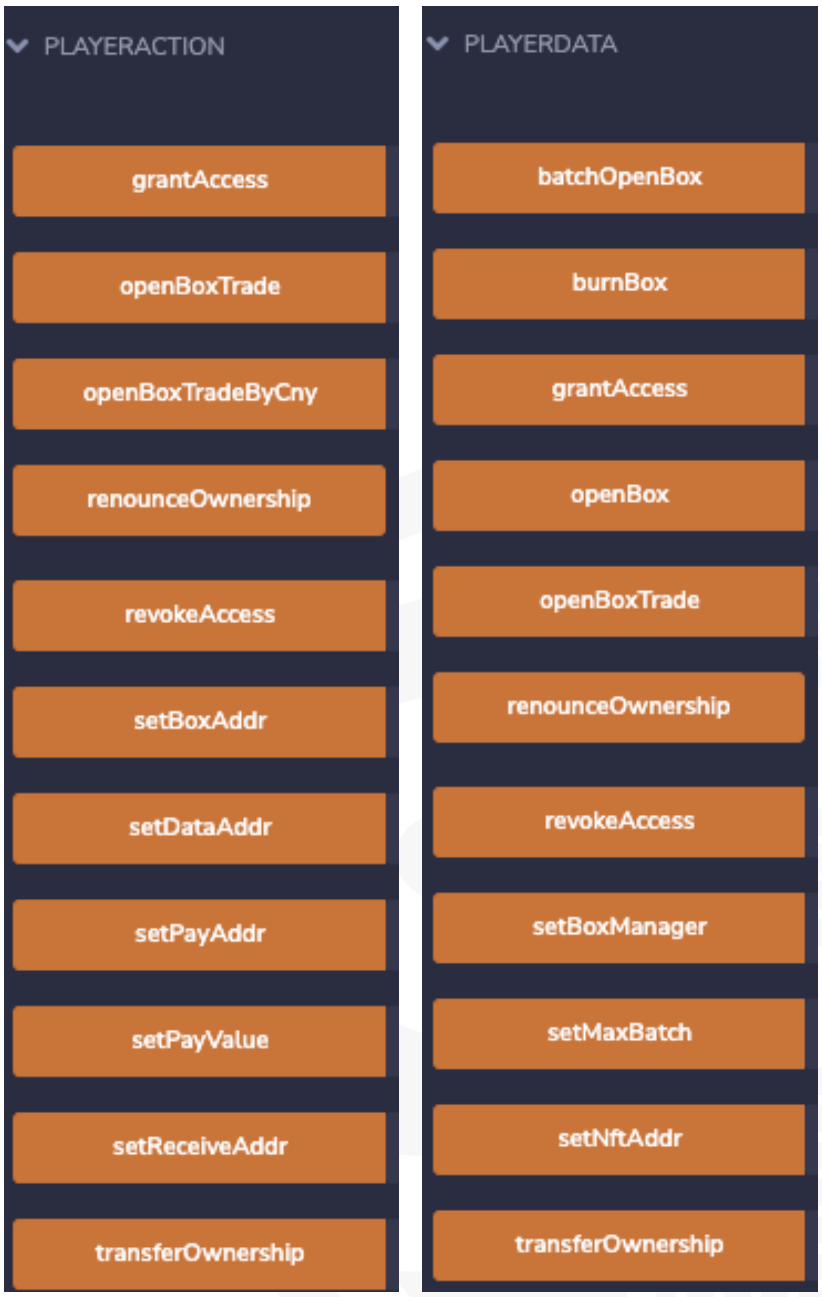
Check functions
[✓] balanceOf(address) is present
    [✓] balanceOf(address) -> () (correct return value)
    [✓] balanceOf(address) is view
[✓] ownerOf(uint256) is present
    [✓] ownerOf(uint256) -> () (correct return value)
    [✓] ownerOf(uint256) is view
[✓] safeTransferFrom(address,address,uint256,bytes) is present
    [✓] safeTransferFrom(address,address,uint256,bytes) -> () (correct return type)
    [✓] Transfer(address,address,uint256) is emitted
[✓] safeTransferFrom(address,address,uint256) is present
    [✓] safeTransferFrom(address,address,uint256) -> () (correct return type)
    [✓] Transfer(address,address,uint256) is emitted
[✓] transferFrom(address,address,uint256) is present
    [✓] transferFrom(address,address,uint256) -> () (correct return type)
    [✓] Transfer(address,address,uint256) is emitted
[✓] approve(address,uint256) is present
    [✓] approve(address,uint256) -> () (correct return type)
    [✓] Approval(address,address,uint256) is emitted
[✓] setApprovalForAll(address,bool) is present
    [✓] setApprovalForAll(address,bool) -> () (correct return type)
    [✓] ApprovalForAll(address,address,bool) is emitted
[✓] getApproved(uint256) is present
    [✓] getApproved(uint256) -> () (correct return value)
    [✓] getApproved(uint256) is view
[✓] isApprovedForAll(address,address) is present
    [✓] isApprovedForAll(address,address) -> () (correct return value)
    [✓] isApprovedForAll(address,address) is view
[✓] supportsInterface(bytes4) is present
    [✓] supportsInterface(bytes4) -> () (correct return value)
    [✓] supportsInterface(bytes4) is view
[✓] name() is present
    [✓] name() -> () (correct return value)

[✓] name() is view
[✓] symbol() is present
[✓] symbol() -> () (correct return value)
[✓] tokenURI(uint256) is present
[✓] tokenURI(uint256) -> () (correct return value)

Check events
[✓] Transfer(address,address,uint256) is present
[✓] parameter 0 is indexed
[✓] parameter 1 is indexed
[✓] parameter 2 is indexed
[✓] Approval(address,address,uint256) is present
[✓] parameter 0 is indexed
[✓] parameter 1 is indexed
[✓] parameter 2 is indexed
[✓] ApprovalForAll(address,address,bool) is present
[✓] parameter 0 is indexed
[✓] parameter 1 is indexed

Write functions of contract


IDOACTION
buy
grantAccess
renounceOwnership
revokeAccess
setChangeAmount
setEndTime
setIdoAmount
setLargestNumber
setMaxtNumber
setMinimumQuantity
setProjectParty
setStartTime
setTokenAddr
transferOwnership
NFTSTAKE
changeIsDis
extraProfit
grantAccess
onERC721Received
pledgeNft
renounceOwnership
revokeAccess
setblockPerNumber
setBox
setInitAddress
setLastBlockNumber
setNftPower
setPlayDataAddr
takeProfit
transferOwnership
unpledgeNft
NFTSTAKEACTION
grantAccess
pledgeNft
renounceOwnership
revokeAccess
setnftAddr
setstakeAddr
takeProfit
transferOwnership
unpledgeNft


PLAYERACTION
grantAccess
openBoxTrade
openBoxTradeByCny
renounceOwnership
revokeAccess
setBoxAddr
setDataAddr
setPayAddr
setPayValue
setReceiveAddr
transferOwnership
PLAYERDATA
batchOpenBox
burnBox
grantAccess
openBox
openBoxTrade
renounceOwnership
revokeAccess
setBoxManager
setMaxBatch
setNftAddr
transferOwnership

Deployer cannot mint any new tokens

Name	Exist	Tested	Verified
cannot mint	✓	✓	✗

Comments:

v1.0

- YuanNft
  - OnlyAccessed addresses can mint with createPlayer function

Deployer cannot burn or lock user funds

Name	Exist	Tested	Verified
cannot lock	✓	✓	✗
cannot burn	✓	✓	✗

Comments:

v1.0

- YuanNft
  - OnlyAccessed addresses can burn with burn function
- boxSale
  - onlyAccessed can lock following functions
    - pledgeNft
- nftStakeAction
  - onlyOwner can lock following functions by activate pause
    - takeProfit
    - pledgeNft
    - unpledgeNft
- PlayerAction
  - onlyOwner can lock following functions by activate pause
    - openBoxTrade

Deployer cannot pause the contract

Name	Exist	Tested	Verified
Deployer cannot pause	✓	✓	✗

Comments:

v1.0

- NftStakeAction
  - Deployer can pause following functions
    - pledgeNft
    - unpledgeNft
    - takeProfit
- PlayerAction
  - Deployer can pause following functions
    - openBoxTrade
- PlayerDatas
  - Library implemented but wasn't used

Overall checkup (Smart Contract Security)

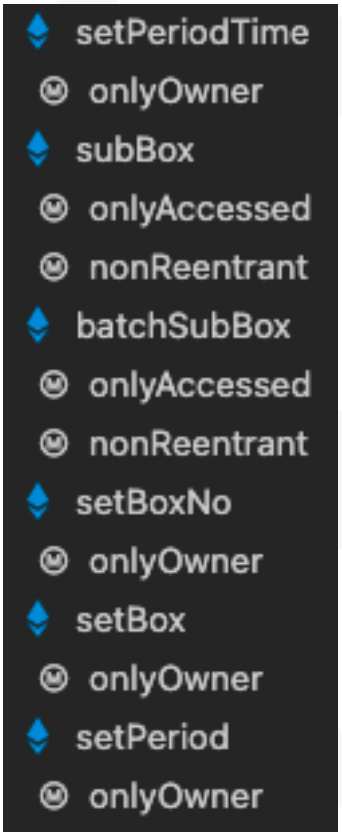
Tested	Verified
✓	✓

Legend

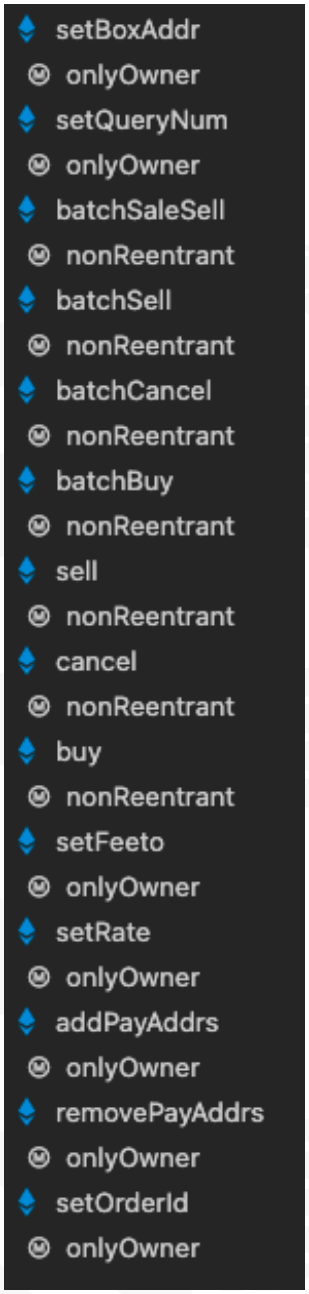
Attribute	Symbol
Verfified / Checked	✓
Partly Verified	⚑
Unverified / Not checked	✗
Not available	–

Modifiers

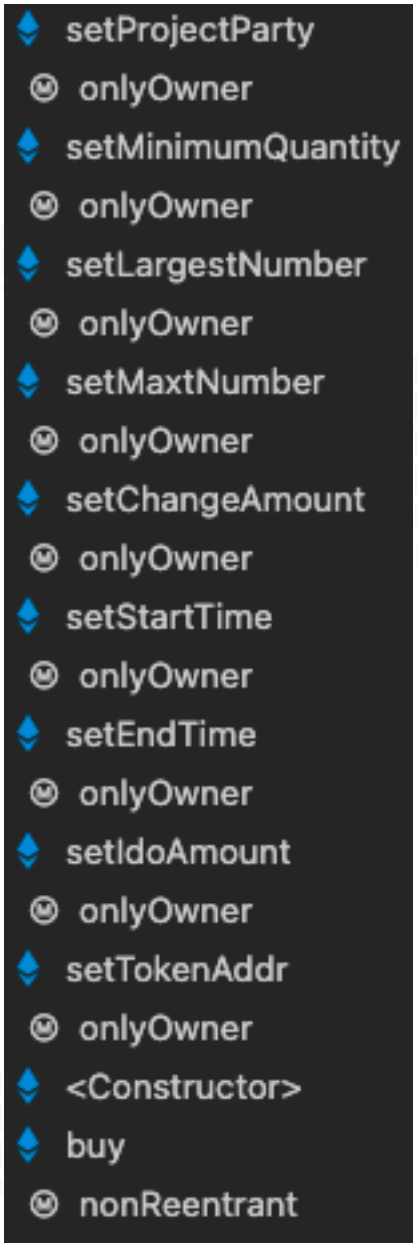
Box



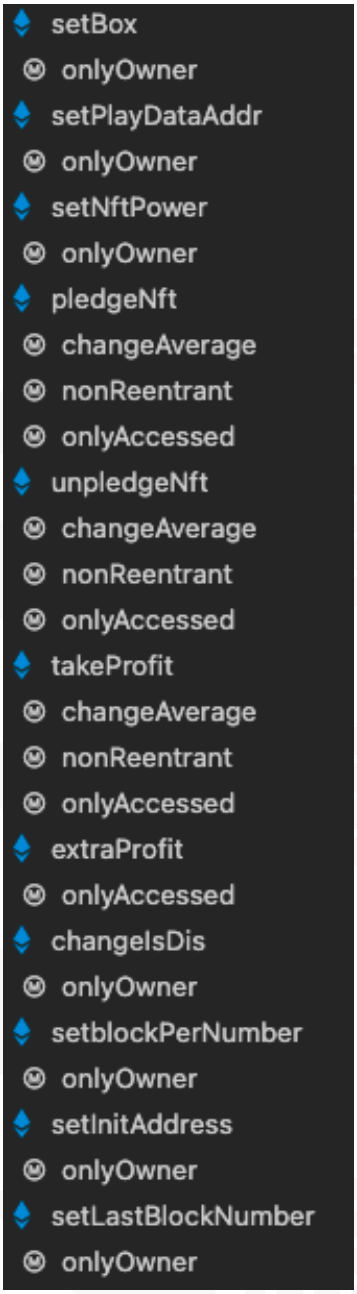
BoxSale



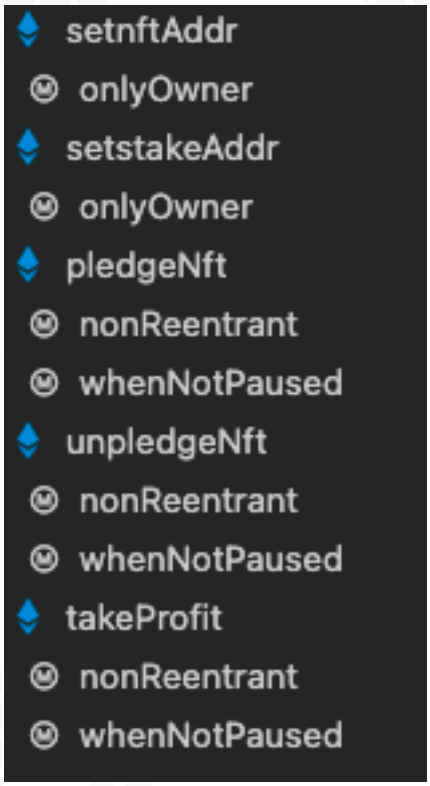
IdoAction



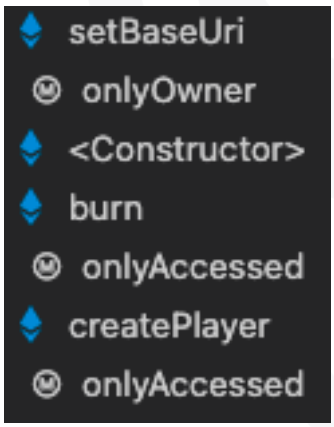
NftStake



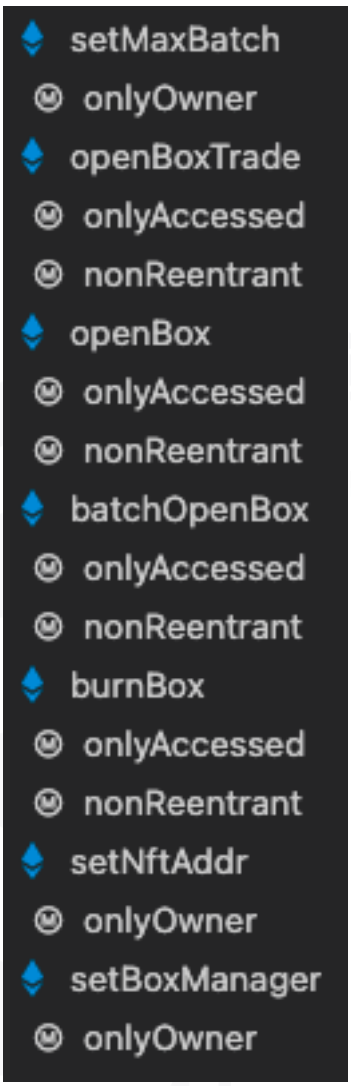
NftStakeAction



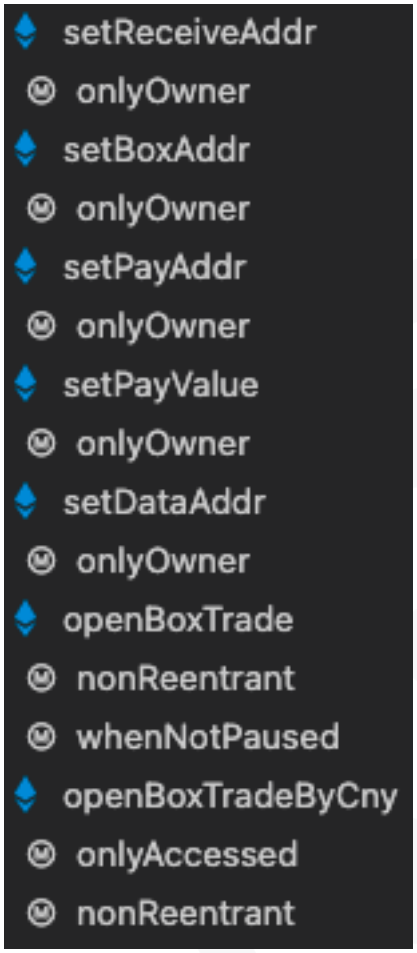
Nft



PlayerDatas



PlayerAction



Comments

- Deployer can set following state variables without any limitations
  - Box
    - period
  - BoxSale
    - queryNum
      - Can only be set higher than previous query num
    - feeTo
    - rateBase
    - rate
    - OrderId
      - Can only be set higher than previous OrderId
  - IdoAction
    - minimumQuantity

        - largestNumber
        - maxtNumber
        - changeAmount
        - startTime
        - endTime
        - idoAmount
    - NftStake
        - _isDIS
        - blockPerNumber
        - lastBlockNumber
    - PlayerAction
        - payValue
    - PlayerDatas
    - maxBatch

- Deployer can enable/disable following state variables
    - Box
        - periodTimes
        - BoxAttrs
        - BoxManagers
        - accessAllowed
    - BoxSale
        - sellers
        - OrderIng
        - BoxSaleOrder
        - Ordered
        - selleds
        - PayAddrs
        - accessAllowed
    - IdoAction
        - accessAllowed
    - Nft
        - accessAllowed
    - NftStake
        - nftPower
        - isSettlement
        - accessAllowed
    - PlayerAction
        - accessAllowed

CallGraph

Source Units in Scope
v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝📚🔍🎨	contracts/nftstake.sol	9	6	501	423	353	4	310	💻💰👥🔆
📝📚🔍🎨	contracts/nftstakeAction.sol	8	4	266	231	187	3	190	💻👥🔆
📝📚🎨	contracts/box.sol	8	————	472	457	356	4	216	💻👥🔆
📝📚🔍🎨	contracts/playerAction.sol	8	3	364	324	261	3	202	💻👥🔆
📝📚🔍🎨	contracts/IdoAction.sol	7	1	350	324	253	4	185	💻👥🔆
📝📚🔍🎨	contracts/nft.sol	11	6	627	561	424	8	409	💻👥🧱🔆♻️
📝📚🔍🎨	contracts/boxSale.sol	9	8	1341	1167	575	633	432	💻📝👥🔆
📝📚🔍🎨	contracts/playerDatas.sol	10	3	586	526	419	4	304	💻👥🧱🔆
📝📚🔍🎨	**Totals**	**70**	**31**	**4507**	**4013**	**2828**	**663**	**2248**	💻📝💰👥🧱🔆♻️

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

High issues

No high issues

Medium issues

No medium issues

Low issues

Issue	File	Type	Line	Description
#1	Main	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	NftStake	State variable visibility is not set	331, 338, 346, 348, 466	It is best practice to set the visibility of state variables explicitly

Informational issues

Issue	File	Type	Line	Description
#1	Box	Naming convention	See description Lines	Use mixedCase naming convention in local variables Recommendation: - lastvalue to lastValue Lines (250, 251, 252) If you want to change variables, make sure to change it everywhere else too
#2	BoxSale	Naming convention	See description Lines	Use mixedCase naming convention in local variables Recommendation: - lastvalue to lastValue Lines (672, 675, 677) If you want to change variables, make sure to change it everywhere else too
#3	PlayerDatas	Naming convention	See description Lines	Use mixedCase naming convention in local variables Recommendation: - lastvalue to lastValue Lines (292, 294, 295) If you want to change variables, make sure to change it everywhere else too
#4	All	Missing require statements message	-	Provide an error message for require statements

Audit Comments

17. January 2022:

- Contract with address 0x1416e6EA40CBb1F09Cd2dbEdAAd6fbFE3e38D51F was not provided to Solidproof, please do your own research here
- Read whole report for more information

20. January 2022:

- PlayerAction
  - New function (mintBoxTrade) were added
- PlayerDatas
  - New function (mintBox) were added

08. February 2022:

- Several issues has been fixed by the team of UniverseUN
- Read whole report 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	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	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

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY