



SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

v1.0: 17. January, 2022

Audit

Security Assessment
20. January, 2022

For



UNIVERSE

LAUNCH PLATFORM

Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Link	5
Methodology	7
Used Code from other Frameworks/Smart Contracts (direct imports)	8
Tested Contract Files	11
Source Lines	12
Risk Level	12
Capabilities	13
Scope of Work	15
Inheritance Graph	15
Verify Claims	16
Modifiers	25
CallGraph	29
Source Units in Scope	30
Critical issues	31
High issues	31
Medium issues	31
Low issues	31
Informational issues	32
Commented Code exist	35
Audit Comments	36
SWC Attacks	37

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	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary
1.1	20. January 2022	<ul style="list-style-type: none">• 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

- 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 as possible.
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

IPlayer
IPlayerInfo
IPay
IERC20
Context
Ownable
Accessible
SafeMath
Address
EnumerableSet
ReentrancyGuard
IERC721Receiver
ERC721Holder
IERC165
IERC721
ISale

Box

Context
Ownable
Accessible
SafeMath
Address
EnumerableSet
ReentrancyGuard

IdoAction

Context
IERC20
Ownable
Accessible
SafeMath
Address
ReentrancyGuard

Nft

IERC721Receiver
INft
ERC721Holder
IERC165
IERC721
IERC721Metadata
IERC721Enumerable
Address
Context
Strings
ERC165
ERC721
ERC721Enumerable
Counters
Ownable
Accessible

NFTStake

SafeMath
IERC165
IERC721
SafeERC20
IERC20
Address
Context
Ownable
Accessible
ReentrancyGuard
IERC721Receiver
ERC721Holder
IBox
IPlayData

NFTStakeAction

Context
Ownable
Accessible
Address
ReentrancyGuard
Pausable
IERC165
IERC721
IERC721Receiver
ERC721Holder
INftStake

PlayerDatas

PlayerAction

IERC20
Context
Ownable
Accessible
📦 SafeMath
📦 Address
ReentrancyGuard
Pausable
IPlayerData
IBox

IBox
INft
IERC20
Context
Ownable
Accessible
📦 SafeMath
📦 Address
📦 EnumerableSet
ReentrancyGuard
Pausable
📦 Counters

Tested Contract Files

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

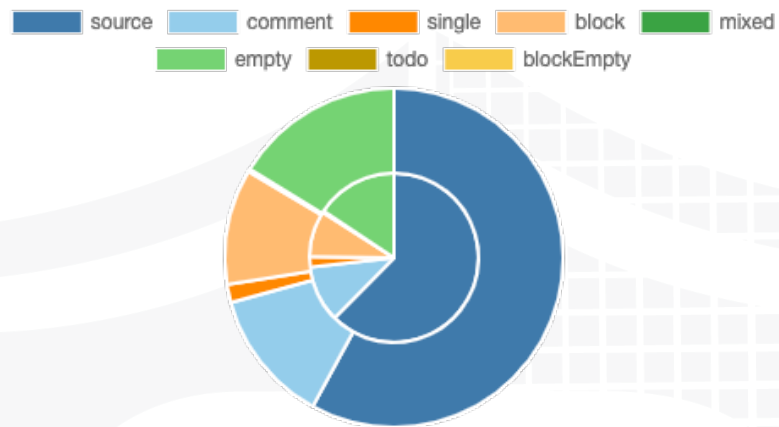
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

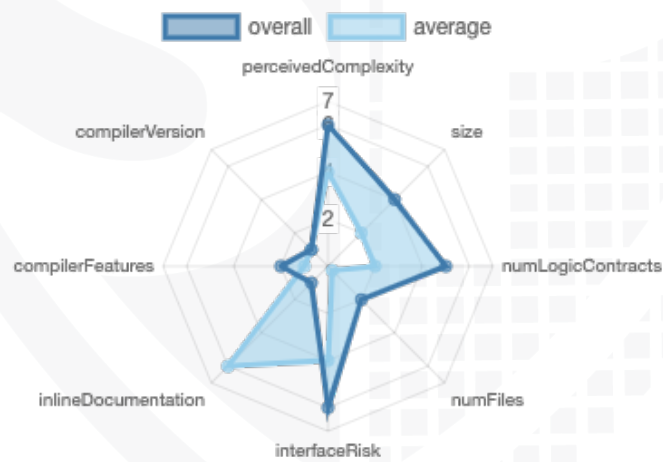
File Name	SHA-1 Hash
contracts/nftstake.sol	4d59ce000f06a70a684cd736ec0ec3357326b3b0
contracts/nftstakeAction.sol	dd36b7f2e6a217d0d58fbc98cf6acc0829492180
contracts/box.sol	363b42391d288e7ae60cc38ead989ee7affc3ce0
contracts/playerAction.sol	345dc07638b055d8316439ca9eb91dbae9b9d135
contracts/ldoAction.sol	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	<code>^0.8.0</code>	ABIEncoderV2	yes	yes (17 asm blocks)	

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

1.0			yes	yes		
-----	--	--	-----	-----	--	--



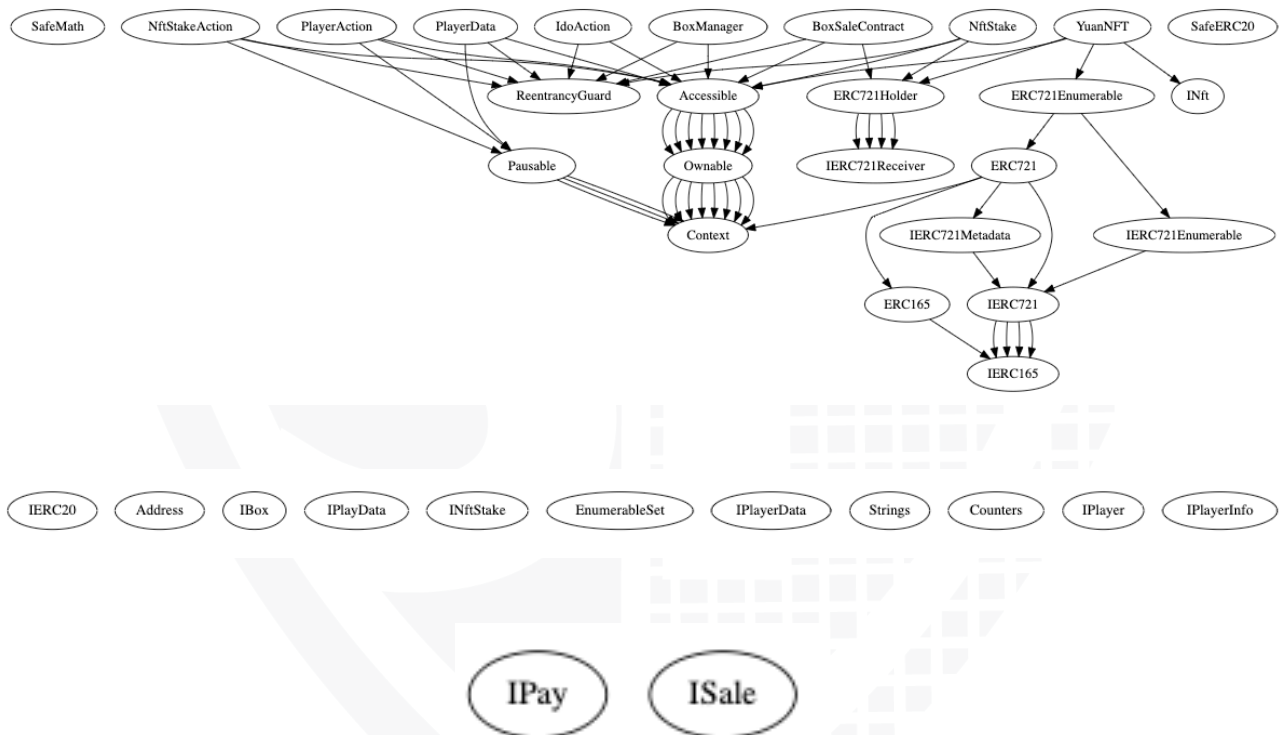
Scope of Work

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

BOXMANAGER	BOXSALECONTRACT	YUANNFT
batchSubBox	addPayAddr	approve
grantAccess	batchBuy	burn
renounceOwnership	batchCancel	createPlayer
revokeAccess	batchSaleSell	grantAccess
setBox	batchSell	onERC721Received
setBoxNo	buy	renounceOwnership
setPeriod	cancel	revokeAccess
setPeriodTime	grantAccess	safeTransferFrom
subBox	onERC721Received	safeTransferFrom
transferOwnership	removePayAddr	setApprovalForAll
	renounceOwnership	setBaseUri
	revokeAccess	transferFrom
	sell	transferOwnership
	setBoxAddr	
	setFeeto	
	setOrderId	
	setQueryNum	
	setRate	
	transferOwnership	

▼ IDOACTION
buy
grantAccess
renounceOwnership
revokeAccess
setChangeAmount
setEndTime
setIdoAmount
setLargestNumber
setMaxtNumber
setMinimumQuantity
setProjectParty
setStartTime
setTokenAddr
transferOwnership

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

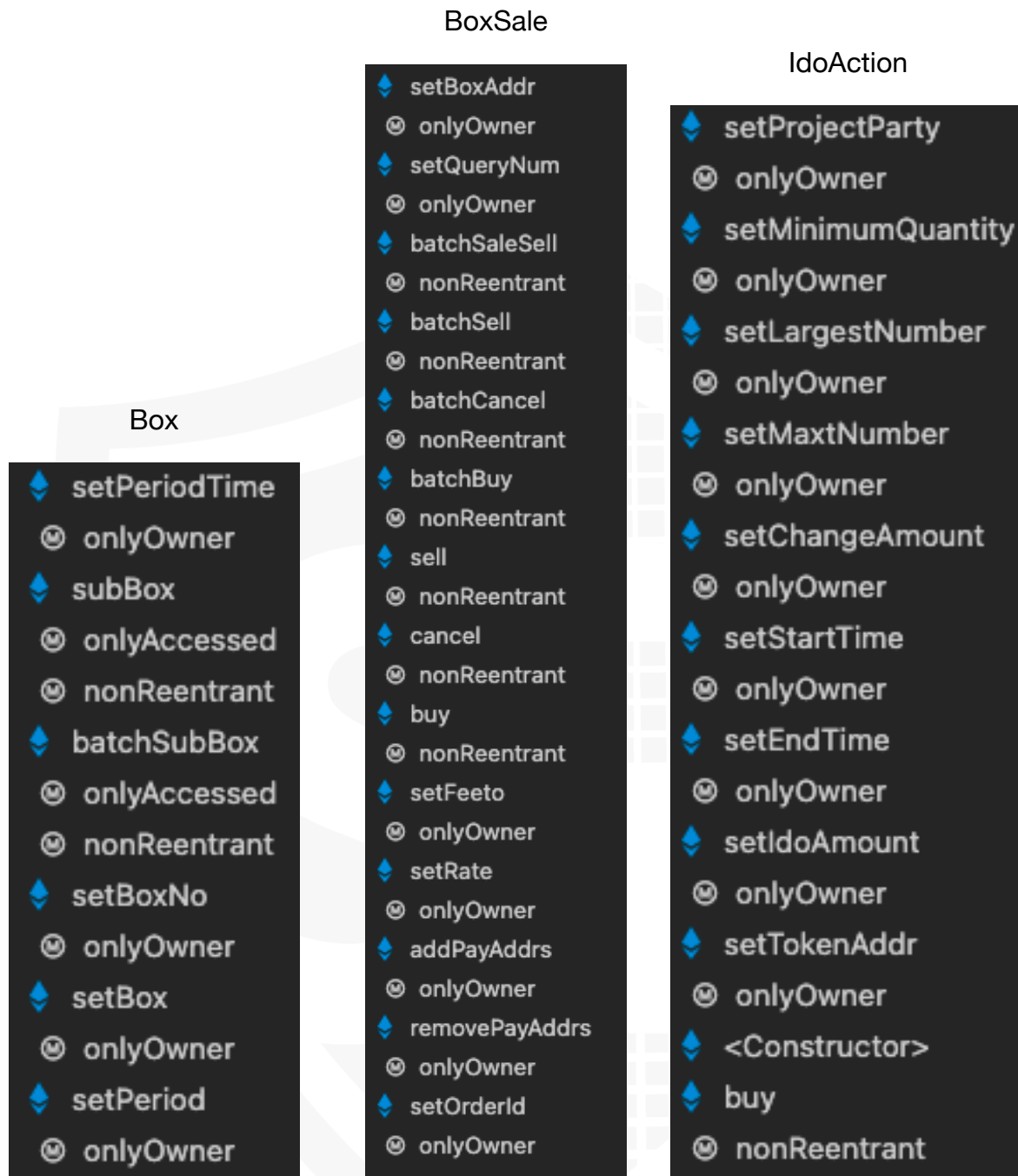
Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

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

Modifiers



NftStake

- ◆ setBox
- Ⓜ onlyOwner
- ◆ setPlayDataAddr
- Ⓜ onlyOwner
- ◆ setNftPower
- Ⓜ onlyOwner
- ◆ pledgeNft
- Ⓜ changeAverage
- Ⓜ nonReentrant
- Ⓜ onlyAccessed
- ◆ unpledgeNft
- Ⓜ changeAverage
- Ⓜ nonReentrant
- Ⓜ onlyAccessed
- ◆ takeProfit
- Ⓜ changeAverage
- Ⓜ nonReentrant
- Ⓜ onlyAccessed
- ◆ extraProfit
- Ⓜ onlyAccessed
- ◆ changelsDis
- Ⓜ onlyOwner
- ◆ setblockPerNumber
- Ⓜ onlyOwner
- ◆ setInitAddress
- Ⓜ onlyOwner
- ◆ setLastBlockNumber
- Ⓜ onlyOwner

NftStakeAction

- ◆ setnftAddr
- Ⓜ onlyOwner
- ◆ setstakeAddr
- Ⓜ onlyOwner
- ◆ pledgeNft
- Ⓜ nonReentrant
- Ⓜ whenNotPaused
- ◆ unpledgeNft
- Ⓜ nonReentrant
- Ⓜ whenNotPaused
- ◆ takeProfit
- Ⓜ nonReentrant
- Ⓜ whenNotPaused

Nft

- ◆ setBaseUri
- Ⓜ onlyOwner
- ◆ <Constructor>
- ◆ burn
- Ⓜ onlyAccessed
- ◆ createPlayer
- Ⓜ onlyAccessed

PlayerDatas

PlayerAction

- ◆ setReceiveAddr
 - Ⓜ onlyOwner
- ◆ setBoxAddr
 - Ⓜ onlyOwner
- ◆ setPayAddr
 - Ⓜ onlyOwner
- ◆ setPayValue
 - Ⓜ onlyOwner
- ◆ setDataAddr
 - Ⓜ onlyOwner
- ◆ openBoxTrade
 - Ⓜ nonReentrant
 - Ⓜ whenNotPaused
- ◆ openBoxTradeByCny
 - Ⓜ onlyAccessed
 - Ⓜ nonReentrant

- ◆ setMaxBatch
 - Ⓜ onlyOwner
- ◆ openBoxTrade
 - Ⓜ onlyAccessed
 - Ⓜ nonReentrant
- ◆ openBox
 - Ⓜ onlyAccessed
 - Ⓜ nonReentrant
- ◆ batchOpenBox
 - Ⓜ onlyAccessed
 - Ⓜ nonReentrant
- ◆ burnBox
 - Ⓜ onlyAccessed
 - Ⓜ nonReentrant
- ◆ setNftAddr
 - Ⓜ onlyOwner
- ◆ setBoxManager
 - Ⓜ onlyOwner

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
 - solds
 - 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/ldoAction.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	BoxSale	Missing Zero Address Validation (missing-zero-check)	1175, 1185, 1353	Check that the address is not zero
#3	IdoAction	Missing Zero Address Validation (missing-zero-check)	315, 279, 311	Check that the address is not zero
#4	NftStake	Missing Zero Address Validation (missing-zero-check)	357, 366, 488, 370	Check that the address is not zero
#5	NftStakeAction	Missing Zero Address Validation (missing-zero-check)	238, 242, 245	Check that the address is not zero

#6	PlayerAction	Missing Zero Address Validation (missing-zero-check)	318, 319, 320, 321, 331, 341, 335,328	Check that the address is not zero
#7	PlayerD atas	Missing Zero Address Validation (missing-zero-check)	487, 599, 595	Check that the address is not zero
#8	NftStake	State variable visibility is not set	322, 331, 338, 346, 348, 466	It is best practice to set the visibility of state variables explicitly
#9	NFT	Local variables shadowing	596	Rename the local variables that shadow another component
#10	Box	Missing Events Arithmetic	464	Emit an event for critical parameter changes
#11	BoxSale	Missing Events Arithmetic	1372, 1189, 1357	Emit an event for critical parameter changes
#12	NftStake	Missing Events Arithmetic	492, 485	Emit an event for critical parameter changes
#13	PlayerD atas	Missing Events Arithmetic	492	Emit an event for critical parameter changes

Informational issues

Issue	File	Type	Line	Description
#1	PlayerD atas	State variables that could be declared constant (constable-states)		Add the `constant` attributes to state variables that never change
#2	Box	Unused return values	459	Ensure that all the return values of the function calls are used and handle both success and failure cases if needed by the business logic
#3	BoxSale	Unused return values	1364, 1275, 1276, 1284, 1285, 1263, 1265, 1266, 1176, 1368, 1254, 1253	Ensure that all the return values of the function calls are used and handle both success and failure cases if needed by the business logic

#4	PlayerAction	Unused return values	363	Ensure that all the return values of the function calls are used and handle both success and failure cases if needed by the business logic
#5	NftStake	Unused state variables	322	Remove unused state variables
#6	PlayerData	Unused state variables	469	Remove unused state variables
#7	NftStake	Unnecessary brackets	490	You can remove brackets from nftDataAddress
#8	Box	Naming convention	See description Lines	<p>Use mixedCase naming convention in local variables</p> <p>Recommendation:</p> <ul style="list-style-type: none"> - lastvalue to lastValue Lines (250, 251, 252) <p>If you want to change variables, make sure to change it everywhere else too</p>
#9	BoxSale	Naming convention	See description Lines	<p>Use mixedCase naming convention in local variables</p> <p>Recommendation:</p> <ul style="list-style-type: none"> - lastvalue to lastValue Lines (672, 675, 677) - _feeto to _feeTo Lines (1354,1353) <p>If you want to change variables, make sure to change it everywhere else too</p>

#10	IdoAction	Naming convention	See description Lines	<p>Use mixedCase naming convention in local variables</p> <p>Recommendation:</p> <ul style="list-style-type: none"> - maxtNumber to maxNumber Lines (264, 292, 319, 334) - getnum to getNum Lines (275, 340) <p>If you want to change variables, make sure to change it everywhere else too</p>
#11	NftStake	Naming convention	See description Lines	<p>Use mixedCase naming convention in local variables</p> <p>Recommendation:</p> <ul style="list-style-type: none"> - allPofit to allProfit Lines (446, 458) <p>If you want to change variables, make sure to change it everywhere else too</p>
#12	NftStake	Naming convention	See description Lines	<p>Use mixedCase naming convention in local variables</p> <p>Recommendation:</p> <ul style="list-style-type: none"> - setnftAddr to setNftAddress Lines (242) - setstakeAddr to setStakeAddress Lines (245) <p>If you want to change variables, make sure to change it everywhere else too</p>

#13	PlayerD atas	Naming convention	See description Lines	<p>Use mixedCase naming convention in local variables</p> <p>Recommendation: - lastvalue to lastValue Lines (292, 294, 295)</p> <p>If you want to change variables, make sure to change it everywhere else too</p>
-----	-----------------	-------------------	-----------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Commented Code exist

There are some instances of code being commented out in the following files that should be removed:

File	Line	Comment
BoxSale	1177, 1178	// PayAddr.add(0x38e8a525c03d3dD6648012221F57e88C7E29CfdC);

Recommendation

Remove the commented code, or address them properly.

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



SWC Attacks

ID	Title	Relationships	Status
SW C-1 36	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SW C-1 35	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SW C-1 34	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SW C-1 33	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SW C-1 32	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SW C-1 31	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SW C-1 30	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SW C-1 29	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SW C-1 28	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SW C-1 27	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SW C-1 25	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SW C-1 24	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SW C-1 23	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SW C-1 22	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SW C-1 21	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SW C-1 20	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SW C-11 9	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	NOT PASSED
SW C-11 8	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SW C-11 7	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SW C-11 6	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 5	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SW C-11 4	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SW C-11 3	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SW C-11 2	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 1	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SW C-11 0	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SW C-1 09	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SW C-1 08	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	NOT PASSED
SW C-1 07	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SW C-1 06	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SW C-1 05	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SW C-1 04	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SW C-1 03	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	PASSED
SW C-1 02	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SW C-1 01	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SW C-1 00	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

The logo features the words "SolidProof" in a white, handwritten-style script. The "P" is large and stylized, with a long horizontal stroke that extends to the left. The background is a solid blue color with a faint, large shield emblem. The shield has a grid-like pattern on its right side and a solid blue area on its left side.

SolidProof

Blockchain Security | Smart Contract Audits | KYC

A small horizontal bar representing the German flag, with black, red, and gold stripes.

MADE IN GERMANY