SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

BetBoom

Audit

Security Assessment

17.November,2022

For

BetBOOM

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	31.October,2022	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	17.November,2022	• Reaudit

Network
Polygon

Website
https://betboom.io/

Telegram
https://t.me/betboomglobalofficial

Twitter
https://twitter.com/betboom_io

Discord
https://discord.gg/katYxVK9g5

Description

BetBOOM represents a decentralized reform in games.
BetBOOM marks a great foray of Web3.0 into games. Through the model “BET to Earn”, guessing games are no longer simply zero-sum ones, with players enjoying diverse benefits. Bring it on! We will embark on a brand-new path of decentralized games.

Project Engagement

During the 31st of October 2022, **BetBoom** team engaged Solidproof.io to audit the smart contracts that they created. The engagement was technical in nature and focused on identifying the security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo

BetBOOM

Contract Links

v1.0

https://github.com/betboom-eco/betboom/commit/9fc1c9c25860d4873e5e013624fe5b75a36f1d21

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 analyzing 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:

```
../../interfaces/IERC20.sol
../../libraries/SafeMath.sol
../../libraries/SafeERC20.sol
../../types/Ownable.sol
../common/Auth.sol
../../interfaces/ILuckyGame.sol
../../interfaces/ILetDaoSwap.sol
../../libraries/EnumerableSet.sol
../../interfaces/ILetDao.sol
```

```
../../interfaces/IERC20.sol
../../libraries/SafeMath.sol
../../libraries/SafeERC20.sol
../common/Op.sol
../../interfaces/ILuckyPool.sol
../../libraries/Address.sol
../../libraries/EnumerableSet.sol
../../types/ReentrancyGuard.sol
../../interfaces/INFTPool.sol
```

```
../../interfaces/IERC721.sol
../common/Operator.sol
../../interfaces/ILuckyGame.sol
../../interfaces/IERC20.sol
../../libraries/SafeMath.sol
../../libraries/SafeERC20.sol
../../interfaces/IExp.sol
../../libraries/EnumerableSet.sol
../../interfaces/IPlayerNFT.sol
../../interfaces/ILetDao.sol
```

```
../../types/ERC20.sol
../../types/MinterOwned.sol
../../libraries/SafeMath.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.

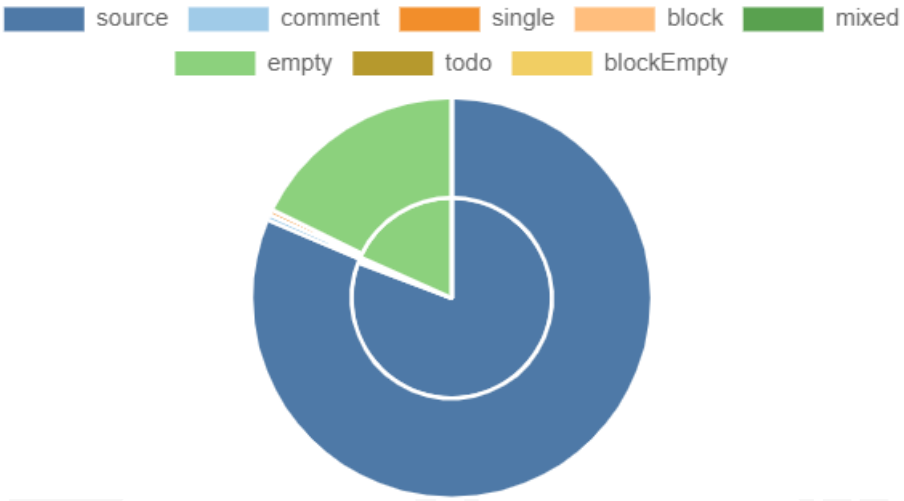
v1.0

File Name	SHA-1 Hash
contracts/games/LuckyGame.sol	635cda402129f59fc36f79eb12779f79d9efaea0
contracts/common/Op.sol	f44bf1e577c5019f89c43d5951236d31c858fc89
contracts/common/Operator.sol	1e9a54568ad75bf28b0ef626f29e79289b36345e
contracts/common/Parama.sol	6370b308988727cd7276601575fd53ed92e1b532
contracts/common/Auth.sol	55ea0f997c7f0a18aac7fbc401f3009aac02614b
contracts/pools/LuckyPool.sol	a6cf7987f0742ae832eec8dca42935b33ba21332
contracts/pools/NFTPool.sol	7aac378302694223177444e4e99bf3d6ca951428
contracts/token/LET.sol	3dbca3af750d5819ce0d169625fe7cfeb9e472dc
contracts/token/BET.sol	07adb2f0dabc148f9c200afd1f841b017fd4707c

Metrics

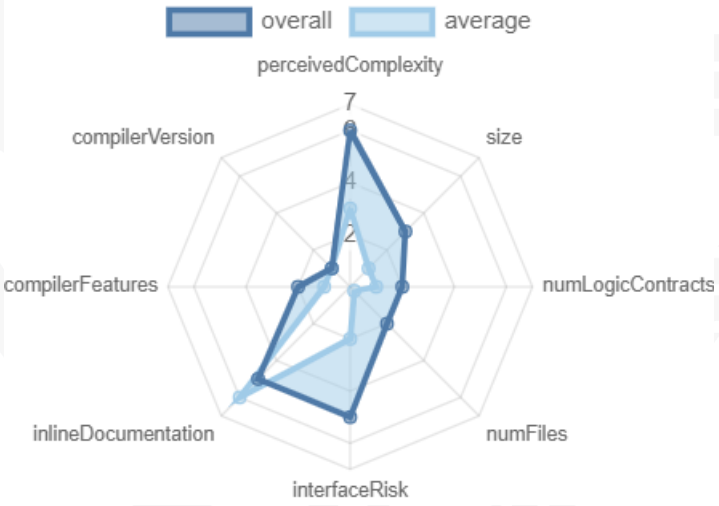
Source Lines

v1.0



Risk Level

v1.0



Capabilities

v1.0

Components

Contracts	Libraries	Interfaces	Abstract
9	0	0	0

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
133	0

External	Internal	Private	Pure	View
95	115	0	7	64

StateVariables

Total	Public
123	103

Capabilities

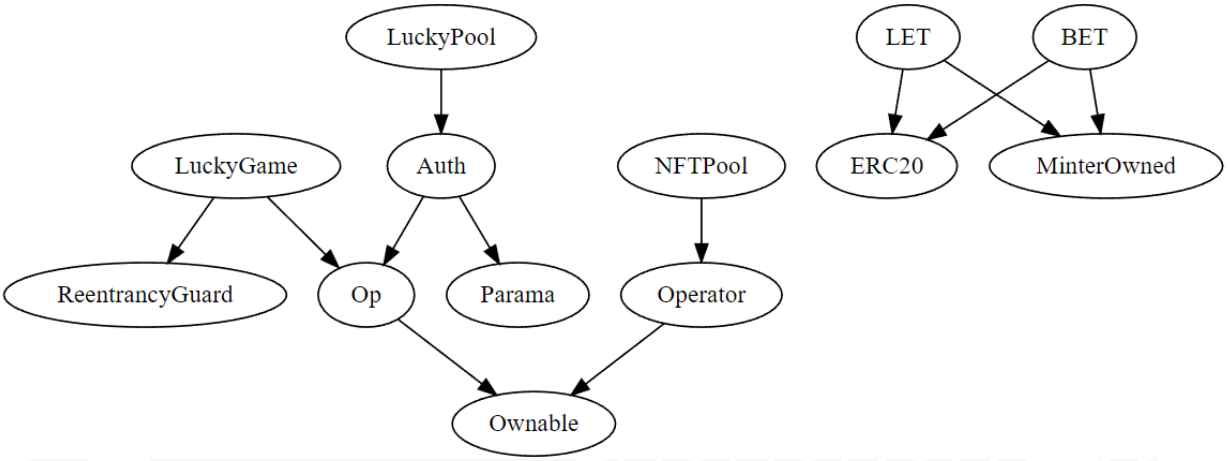
Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
`^0.8.0`	`ABIEncoderV2`	—	—	—

Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/Create/Create2
—	—	—	`yes`	—	—

TryCatch	Σ Unchecked
—	—

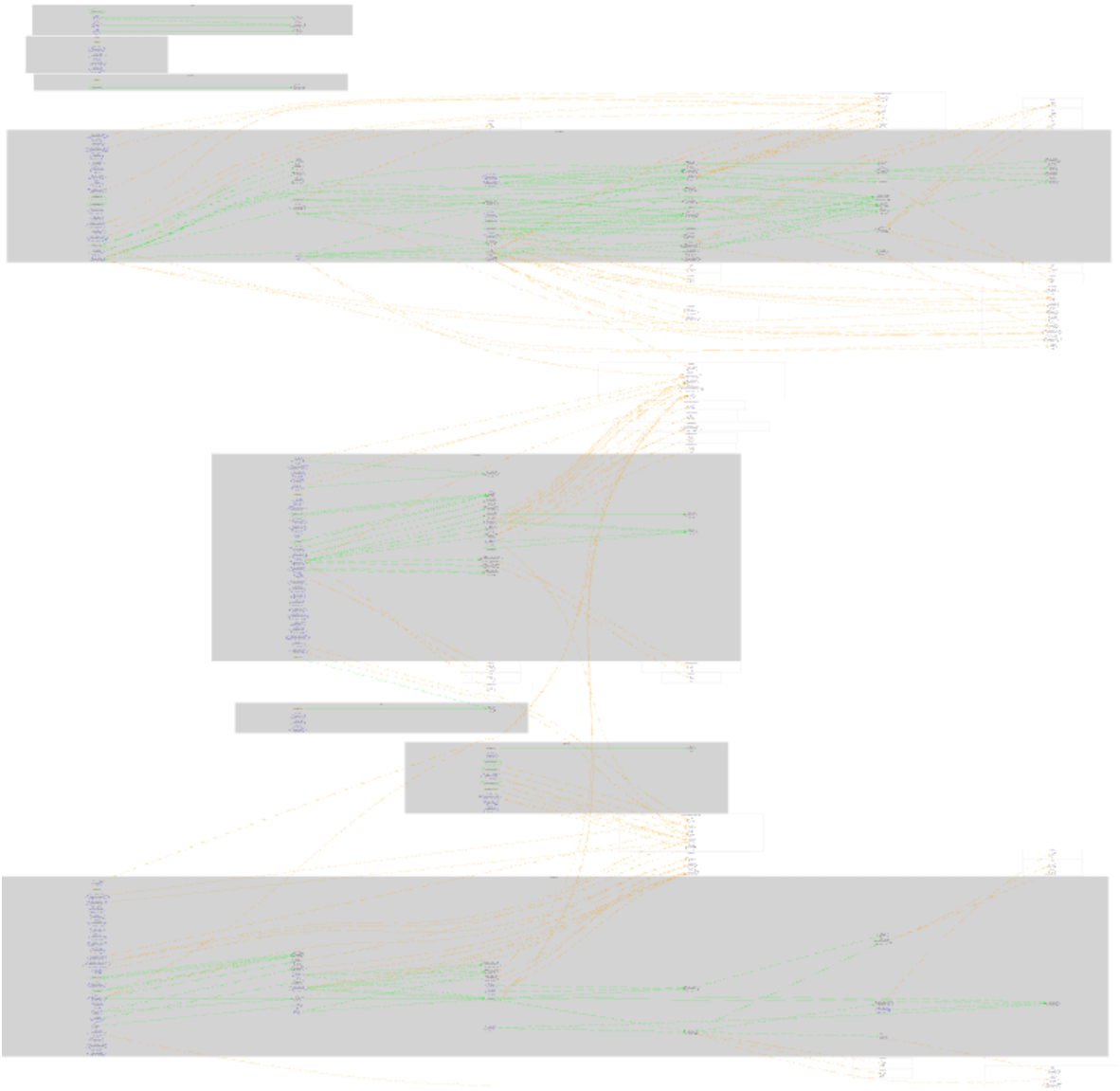
Inheritance Graph

v1.0



Call Graph

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 can set fees
7. Deployer can blacklist/antisnipe address
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			

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer can mint			
Max / Total Supply	N/A		

Comments:

- The owner can mint tokens wherever there is a functionality.
- Multiple authorities can mint tokens. For example, the operator account and there are also “OnlyMinter” accounts that can be set by the operator account.
- Even after renouncing the ownership, if the operator account and other authorities are not set to zero

Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock			
Deployer cannot burn			

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause			

Deployer can set fees

Name	Exist	Tested	Status
Deployer can set fees over 25%			
Deployer can set fees to nearly 100% or more			

Comments:

- The maximum transaction fees can be manipulated by everyone in the FactoryBallGame.
- The owner can set the fees to any number in the RewardPool contract

Deployer cannot blacklist/antisnipe addresses

Name	Exist	Tested	Status
Deployer can blacklist/antisnipe addresses			

Overall checkup (Smart Contract Security)

Tested	Verified

Legend

Attribute	Symbol
Verified / Checked	
Partly Verified	
Unverified / Not checked	
Not available	

Modifiers, public, and Write functions

v1.0

FactoryBallGame

- setBetAmount
- onlyOperator
- notEnd
- claim
- claimMatch
- caculateMatch
- setCancel
- setResult
- setCaculateAccount
- onlyOwner
- bet
- setFeeRate
- setMathIsOpen
- onlyOperator
- addMatch
- addOrRemoveTeam
- onlyOperator
- isExist
- notStart
- setMaxTeam
- onlyOperator
- isExist
- notEnd
- setCanPlay
- onlyOperator
- isExist
- notEnd
- setEndTime
- onlyOperator
- isExist
- notEnd
- setStartTime
- onlyOperator
- isExist
- notStart
- setCupName
- onlyOperator
- isExist
- notStart
- createCup
- addOrRemoveGame
- onlyOperator

RewardPool

- setExplosionAmount
- onlyOperator
- setFactory
- onlyOperator
- onlyAddAmount
- setLockTime
- onlyOperator
- setLPToken
- onlyOperator
- addPoolAmount
- nonReentrant
- checkAmount
- updateReward
- claimInsurer
- nonReentrant
- checkAmount
- updateReward
- claimReward
- updateReward
- updateAmount
- updateCaculateAmount
- updateValue
- cliam
- setWeekDays
- onlyOperator
- update
- setCoe
- onlyOperator
- setMaxNum
- onlyOperator
- setRate
- onlyOperator
- setRank
- setProAccount
- onlyOperator
- setDaoAccount
- onlyOperator
- setLP
- onlyOperator
- setLetDaoSwap
- onlyOperator
- transferTo
- onlyOwner
- addMint
- onlyFactory
- setUpAmount
- onlyOperator
- blast
- onlyFactory
- setRewardAmount
- onlyOperator
- updateWeek
- initBnbTime

LuckyPool

- setRewardAmount
- onlyOperator
- initBnbTime
- setRate
- onlyOperator
- setRank
- setAssetAccount
- setLuckyGame
- setProAccount
- onlyOperator
- setDaoAccount
- onlyOperator
- setLP
- setLetDaoSwap
- transferTo
- onlyOwner
- initAssets
- addAssets
- addBetAmount
- userClaim
- updateWeek
- addMint
- onlyGame
- setExplosionAmount
- onlyOperator

NFTPool

```
setERC721
onlyOperator
setExp
onlyOperator
setLockTime
onlyOperator
setSellAmount
onlyOperator
setMaxSellNum
onlyOperator
setPlayerNFT
onlyOperator
setDaoAccount
onlyOperator
setBonusAccount
onlyOperator
setBuyNum
onlyOperator
mintTo
onlyOperator
buyAndDeposit
updatePool
deposit
withdraw
claim
changeTokenID
increaseMint
onlyContractAuth
gainExperience
onlyContractAuth
castNFT
onlyPlayer
claimNFT
onlyPlayer
updateWeek
mintLET
transferTo
onlyOwner
```

```
setIsOpen
onlyOperator
addOrRemoveWhiteList
onlyOperator
updateUser
onlyPlayer
```

Ownership (and other authorities controlled by the owner) Privileges:

The owner can assign authorities to other accounts and they can call the functions with the modifiers like, onlyGame, onlyPlayer, onlyOperator, onlyMinter, etc.

S.No.	File	Description
#1	Auth.sol	• Set "UP" amount, Add amount, limit amount, and per amount but more than zero
#2	Operator.sol	• Set operator, add/remove auth contracts
#3	FactoryBallGame.sol	• Set result, calculate account, match open, add match, and create cup
#4	LuckyGame.sol	• Can call setBlockHash and bethSetBlockHash functions
#5	LuckyPool.sol	• Set reward amount, rate, rank, asset account, LuckyGame contract address, pro account, DAO Account, LP pair, and initialize BNB time • Transfer tokens • Initialize and Add assets and bet amount • Call the userClaim function • Update week, and change the NFT Pool • Add mint and set explosion amount
#6	RewardPool	• Set explosion amount, factory contract, lock time, LP token, week days, reward amount, MaxNum, Rate, rank,pro account, dao account, and initializeBNB time. • Transfer tokens, add mint, and call the explosion function

#7	NFTPool.sol	• Set ERC721, EXP, LockTime, Sell Amount, Max sell number, Player NFT, DAO Account, Bonus Account, Buy Number, IsOpen • Increase mint by increasing BET and LET amounts • Add/Remove accounts from the whitelist. • Mint LET tokens by the bonus accounts which is controlled by the operator. • Transfer the balance of the contract (including native tokens) to any wallet, and Update user

Source Units in Scope

v1.0

File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/games/LuckyGame.sol	1		840	806	674	1	434
contracts/common/Op.sol	1		29	29	19	1	14
contracts/common/Operator.sol	1		58	58	43	1	38
contracts/common/Parama.sol	1		51	51	41	1	21
contracts/common/Auth.sol	1		30	30	22	1	23
contracts/pools/LuckyPool.sol	1		436	426	345	1	303
contracts/pools/NFTPool.sol	1		492	470	381	1	291
contracts/token/LET.sol	1		43	43	29	1	23
contracts/token/BET.sol	1		38	38	29	1	20
Totals	**9**		**2017**	**1951**	**1583**	**9**	**1167**

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

Issue	File	Type	Line	Description
#1	RewardPool.sol	Fees can be 100%	344	The authorized addresses can set the fees amount to any arbitrary value.
#2	BSBall.sol	Claim will not work	354	The factory contract will not get the claim here, It will only return the amount of the result or getChoiceRate but it will not be able to claim any tokens

Low issues

Issue	File	Type	Line	Description
#1	Op.sol	Missing zero address validation	16	We recommend to check that the passed address is not zero
#2	LuckyPool.sol	Missing Zero address validation	147-170	We recommend to check that the passed address is not zero
#3	Auth.sol	Missing Events	All	Emit events for critical parameter changes

#4	All	Floating Pragma	-	The current pragma Solidity directive is “^0.8.0”. Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using other versions.
#5	LuckyPool.sol	Owner can Drain Tokens	174	Owner can withdraw native tokens because there is no protection against it.
#6	NFTPool.sol	Missing Events	106-133	Emit events for critical parameter changes
#7	NFTPool..sol	Missing Zero address validation	98,102,121-129	We recommend to check that the passed address is not zero
#8	LuckyPool.sol	Missing Events	96,101,109,147-170,413,391	Emit events for critical parameter changes
#9	BSBall.sol	Uninitialized State Variables	148,156	indexMaxTake is never initialized that means that this variable will always be 0 • We recommend to set it this variable after getting the maxTake • beforeTake will always be smaller than afterTake because it is set to 0 all the time (see • above) this causes that the else condition is not reachable
#10	BSBall.sol	Variables Can be set without limitations	173	• concedeInfo[cupID][mID].number can be set without limitation in L182 • Number should not be modulo 25 != 0 • Next number should not be smaller than the previous number in the array • concedeInfo[cupID][mID].payRate in L183 • payRate should not be under 10.000 or should not be higher than uRate • (factory.checkGame()
#11	LuckyPool.sol	Shadowing Local Variables	331	Rename the local variables that shadow another component.
#12	RewardPool.sol	Owner can drain tokens	372	The factory address can drain the contract’s balance.

#13	All	Contract doesn't import npm packages from source (like OpenZeppelin etc.)	-	We recommend importing all packages from npm directly without flattening the contract. Functions could be modified or can be susceptible to vulnerabilities

Informational issues

Issue	File	Type	Line	Description
#1	All	NatSpec documentation missing	–	If you started to comment your code, also comment all other functions, variables etc.
#2	NFTPool.sol	Uninitialized local variable	348	We recommend to initialize all local variables
#3	BSBall.sol	Unnecessary require statement	215	Unnecessary require statement because ".isBet" will never become true on line 221
#4	BSBall.sol	State Variables missing visibility	26-29,35-37,41-42	Make sure to explicitly define visibility of all variables
#5	FactoryBallGame.sol	State Variables missing visibility	45,46,110-114	Make sure to explicitly define visibility of all variables
#6	IRewardPool.sol	Wrong Spelling of functions	9,21	Correct the spellings to improve the readability of the code
#7	Main	Dead Code	-	Unused/Dead/Commented code exists in the contract and we recommend to remove all of it

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information https://docs.soliditylang.org/en/v0.5.10/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.

17. November, 2022:

- There is still an owner (Owner still has not renounced ownership)
- Read the whole report and modifiers section for more information.
- Reward Pool contract was not provided to SolidProof so we cannot comment on the safety of its code.

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	NOT 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-	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED

131			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-	Write to Arbitrary	CWE-123: Write-what-where Condition	PASSED

124	Storage Location		
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	NOT 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	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

SolidProof_io

@solidproof_io

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY