SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Audit

Security Assessment

13. December, 2021

For

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 9
Source Lines 10
Risk Level 10
Capabilities 11
Scope of Work 13
Inheritance Graph 13
Verify Claims 14
Modifiers 21
CallGraph 25
Source Units in Scope 26
Critical issues 27
High issues 27
Medium issues 27
Low issues 27
Informational issues 28
Commented Code exist 29
Audit Comments 29
SWC Attacks 30

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	13. December 2021	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Binance Smart Chain (BEP20)

Website
https://dinoland.io/

Telegram
https://t.me/dinolandglobal

Twitter
https://twitter.com/dinolandgame

Discord
https://discord.com/invite/ujctynMMk3

Description

Welcome to Dinoland.
Let's create, nurture your own dinosaurs to get RICH in this ultimate virtual world.
Join our squad NOW!

Project Engagement

During the 9th of December 2021, **Dinoland Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo

Contract Link

v1.0

- TBA

Vulnerability & Risk Level

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

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

Auditing Strategy and Techniques Applied

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

Methodology

The auditing process follows a routine series of steps:

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

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

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

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

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

Imported packages:

Dependency / Import Path	Count
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol	1
@openzeppelin/contracts/access/Ownable.sol	3
@openzeppelin/contracts/security/Pausable.sol	1
@openzeppelin/contracts/security/ReentrancyGuard.sol	2
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	2
@openzeppelin/contracts/token/ERC721/IERC721.sol	1
@openzeppelin/contracts/utils/Strings.sol	1
@openzeppelin/contracts/utils/math/SafeMath.sol	1

Tested Contract Files

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

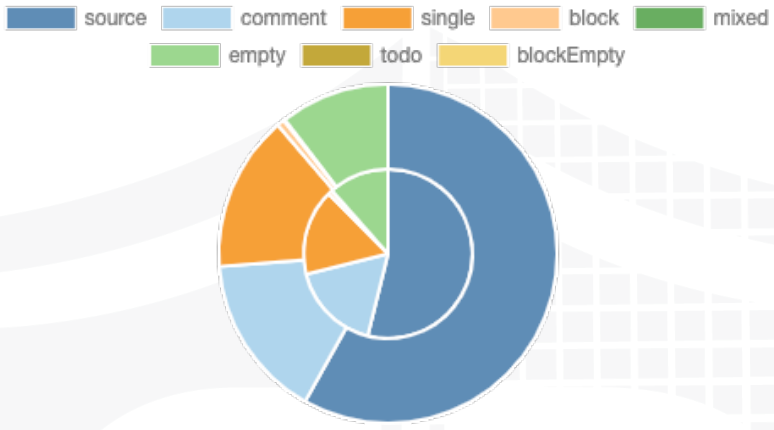
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

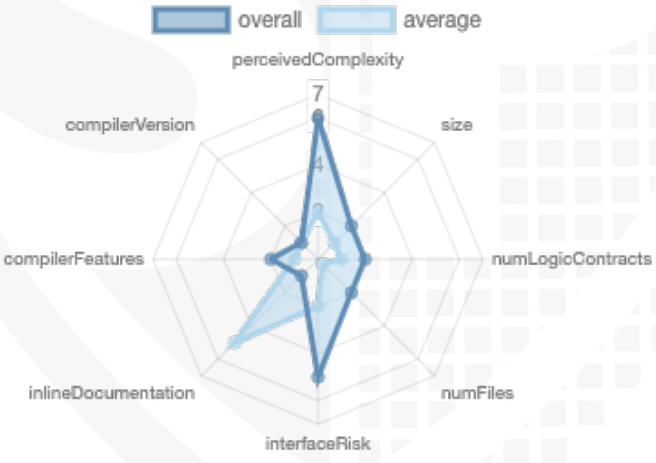
File Name	SHA-1 Hash
contracts/DinoLandNFTUpgradeable.sol	1a716cc65c0160e0dccedb383d5d0e3d6a9a0df3
contracts/DinosAccessControl.sol	91a7adde291f5a5cef6022d0eaf9b21d2b4109f1
contracts/GameManager.sol	96c7b18ef5f2c178bd1a9a4c6a6bcc19dcb2d46e
contracts/DinoMarketplaceUpgradeable.sol	710a79a7e0e6841f4eeca341278160d04cea20ac
contracts/Dinoland.sol	6118d669b686bebab13be3362cac5d9f642e9486
contracts/TokenTimelock.sol	81f149a928f10ce454d3a05b3a28ab6288381be9
contracts/interface/IDinolandNFT.sol	5ffdeb1664e47e0aab2550df94650fa555c3c3f4

Metrics

Source Lines
v1.0



Risk Level
v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	7	0	2	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.

Version	Public	Payable
1.0	76	1

Version	External	Internal	Private	Pure	View
1.0	53	83	3	4	25

State Variables

Version	Total	Public
1.0	59	37

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`0.8.4` `^0.8.0`		`yes`	`yes` (2 asm blocks)	

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

1.0	yes			yes		yes → New Contract:TimelockFactory → AssemblyCall: Name: create → New Contract:TokenTimelock

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

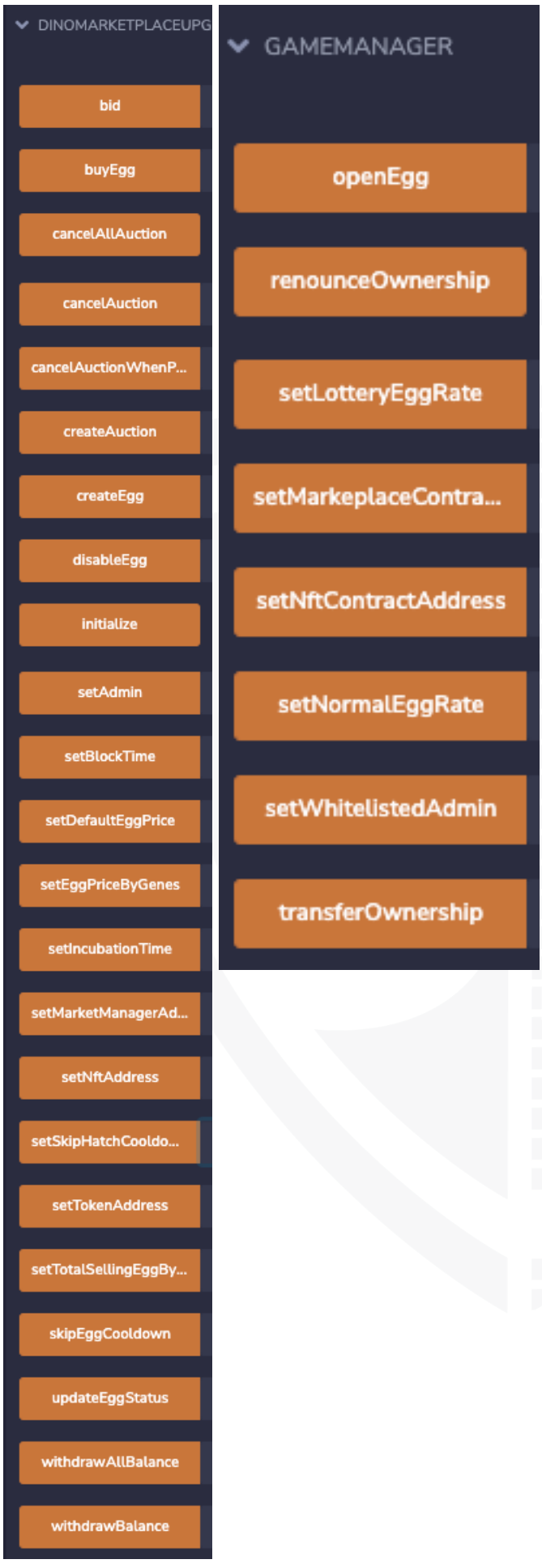
Verify Claims

Correct implementation of Token standard

Tested	Verified
✓	✓

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	✓	✓	✓

Write functions of contract


DINOMARKETPLACEUPG
bid
buyEgg
cancelAllAuction
cancelAuction
cancelAuctionWhenP...
createAuction
createEgg
disableEgg
initialize
setAdmin
setBlockTime
setDefaultEggPrice
setEggPriceByGenes
setIncubationTime
setMarketManagerAd...
setNftAddress
setSkipHatchCooldo...
setTokenAddress
setTotalSellingEggBy...
skipEggCooldown
updateEggStatus
withdrawAllBalance
withdrawBalance
GAMEMANAGER
openEgg
renounceOwnership
setLotteryEggRate
setMarkeplaceContra...
setNftContractAddress
setNormalEggRate
setWhitelistedAdmin
transferOwnership

Deployer cannot mint any new tokens

File	Name	Exist	Tested	Verified
Dinoland	Deployer cannot mint	✓	✓	✗
DinolandNFTUpgradeable	Deployer cannot mint	✓	✓	✓

Max / Total Supply: 1.000.000.000

Comments:

v1.0

- Dinoland
  - onlyOwner can only mint tokens as long as totalSupply() + amount is less than _maxTotalSupply
- DinolandNFTUpgradeable
  - While creating a Dino it has to mint new nft

Deployer cannot burn or lock user funds

File	Name	Exist	Tested	Verified
Dinoland	Deployer cannot lock	✓	✓	✗
DinolandNFTUpgradeable	Deployer cannot burn	✓	✓	✓

Comments:

v1.0

- Dinoland
  - Everybody can burn
- DinolandNFTUpgradeable
  - retireDino can only call whitelisted and onlySpawner

Deployer cannot pause the contract

File	Name	Exist	Tested	Verified
DinolandNFTUpgradeable	Deployer cannot pause	✓	✓	✗

Comments:

v1.0

- Following roles can pause (CLevel)
  - coo
  - Ceo
  - Coo

Overall checkup (Smart Contract Security)

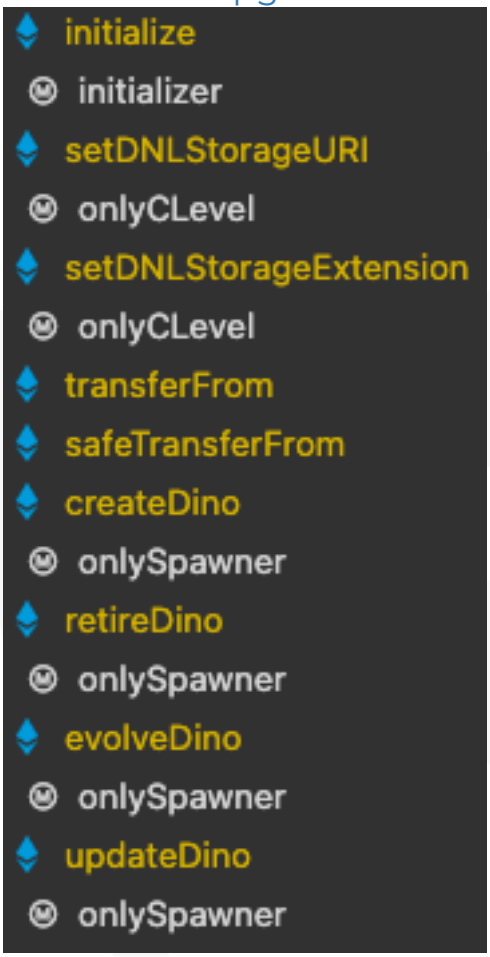
Tested	Verified
✓	✓

Legend

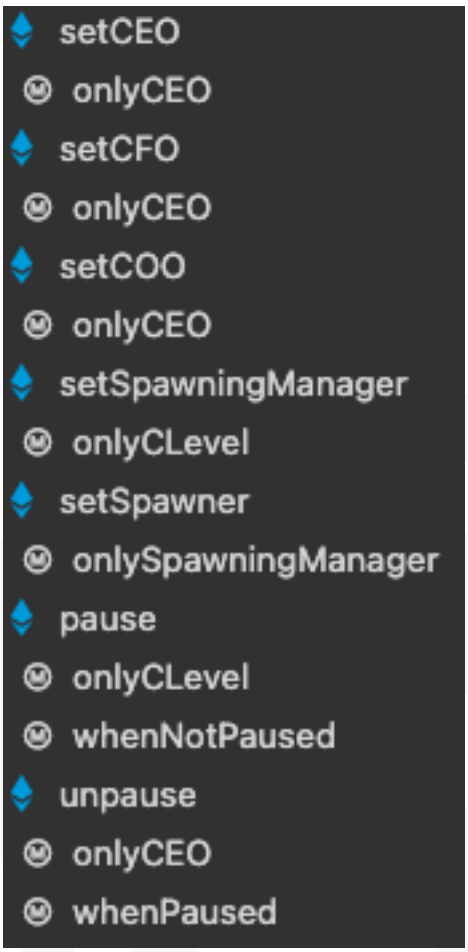
Attribute	Symbol
Verfified / Checked	✓
Partly Verified	⚑
Unverified / Not checked	✗
Not available	–

Modifiers

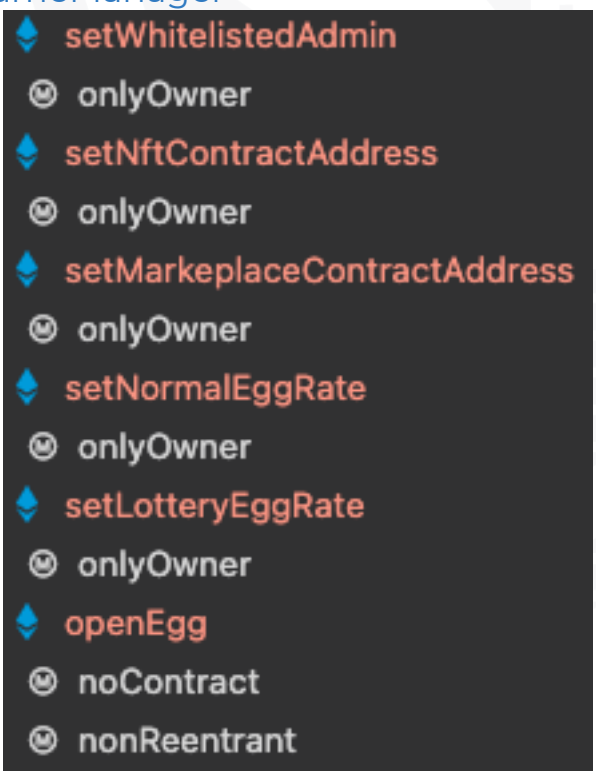
- Dinoland
  - onlyOwner
    - mint
- DinoLandNFTUpgradeable



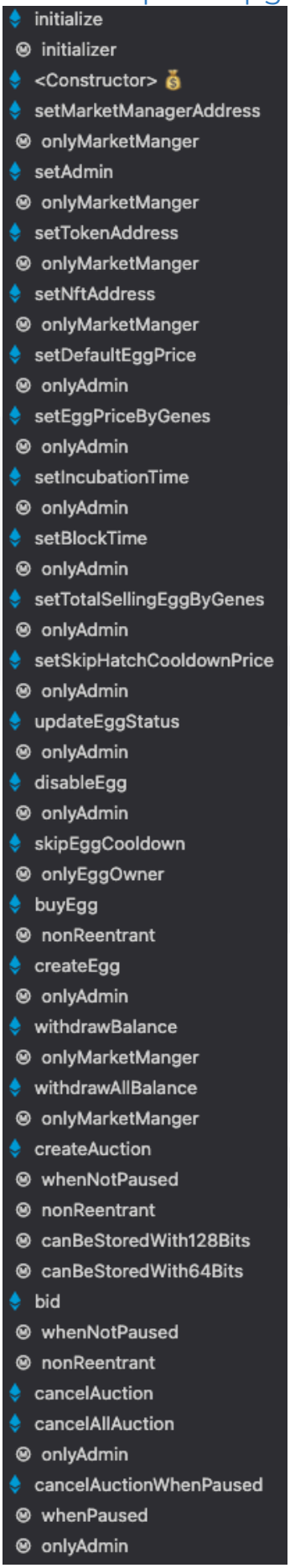
- DinoAccessControl



- GameManager



- DinoMarketplaceUpgradeable



Comments

- Deployer can set following state variables without any limitations
  -
- Deployer can enable/disable following state variables
  -

CallGraph

Source Units in Scope
v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/DinoLandNFTUpgradeable.sol	1	———	313	255	176	58	92	———
🎨	contracts/DinosAccessControl.sol	1	———	128	128	76	28	45	———
📝🔍	contracts/GameManager.sol	1	1	187	183	142	16	94	🖥🧱
📝	contracts/DinoMarketplaceUpgradeable.sol	1	———	792	666	387	198	232	💰💠
📝	contracts/Dinoland.sol	1	———	63	63	48	2	55	🌀
📝	contracts/TokenTimelock.sol	3	———	151	151	115	2	131	🖥💠🌀
🔍	contracts/interface/IDinolandNFT.sol	———	1	7	5	3	1	5	———
📝🔍🎨	**Totals**	**8**	**2**	**1641**	**1451**	**947**	**305**	**654**	🖥💰💠🧱🌀

Legend

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

Audit Results

AUDIT PASSED

Critical issues

- no critical issues found -

High issues

- no high issues found -

Medium issues

Issue	File	Type	Line	Description
#1	GameManager	Weak PRNG	122, 123, 124	There are services to generate random numbers in Smart contracts like Chainlink VRF (For further information read the doc: https://docs.chain.link/docs/chainlink-vrf/) Do not use `block.timestamp`, `now` or `blockhash` as a source of randomness

Low issues

Issue	File	Type	Line	Description
#1	DinosAccessControl	A floating pragma is set	4	The current pragma Solidity directive is „“^0.8.0””.
#2	DinoMarketplaceUpgradeable	Missing Zero Address Validation (missing-zero-check)	159, 184, 178, 90	Check that the address is not zero
#3	GameManager	Missing Zero Address Validation (missing-zero-check)	17, 18, 82, 78	Check that the address is not zero

#4	DinoMarketplaceUpgradeable	Contract locking ether	-	Remove the payable attribute or add a withdraw function
#5	TokenTimelock	Missing Events Arithmetic	120	Emit an event for critical parameter changes
#6	DinoMarketplaceUpgradeable	Missing Events Access Control	159	Emit an event for critical parameter changes
#7	DinolandNFTUpgradeable	Missing Events Access Control	68-75	DinolandNFTUpgradeable.initialize() (DinoLandNFTUpgradeable.sol:68-75) should emit an event for: • ceoAddress = msg.sender (DinoLandNFTUpgradeable.sol#70) • spawningManager = msg.sender (DinoLandNFTUpgradeable.sol#71) Emit an event for critical parameter changes
#8	DinoAccessControl	Missing Events Access Control	70, 78, 85, 90	Emit an event for critical parameter changes
#9	DinolandNFTUpgradeable	State variable visibility is not set	56, 59, 61, 63	It is best practice to set the visibility of state variables explicitly
#10	GameManager	State variable visibility is not set	34, 35, 36, 37, 38, 40, 41, 42, 43, 44	It is best practice to set the visibility of state variables explicitly
#11	DinoMarketplaceUpgradeable	Unchecked tokens transfer	775, 781, 345, 351, 327, 421	Use `SafeERC20`, or ensure that the transfer/ transferFrom return value is checked

Informational issues

- no informational issues found -

Commented Code exist

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

File	Line	Comment
DinoMarketplaceUpgradeable	574	// require(candidateContract.implementsERC721());
	619	// _addAuction())

Recommendation

Remove the commented code, or address them properly.

Audit Comments

13. December 2021:

- 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	CWE-294: Authentication Bypass by Capture-replay	PASSED
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SWC-127	Arbitrary Jump with Function Type	CWE-695: Use of Low-Level Functionality	PASSED
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SWC-120	Weak Sources of Randomness from Chain	CWE-330: Use of Insufficiently Random Values	NOT 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	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

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY