SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Fraktal

Audit

Security Assessment
25. February, 2022

For

Disclaimer

Version	Date	Description
1.0	25. February 2022	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Ethereum Chain

Website
https://www.fraktal.io/

Twitter
https://twitter.com/FraktalNFT

Instagram
https://www.instagram.com/fraktal.io/

Discord
https://discord.com/invite/P6fCPvtZtq
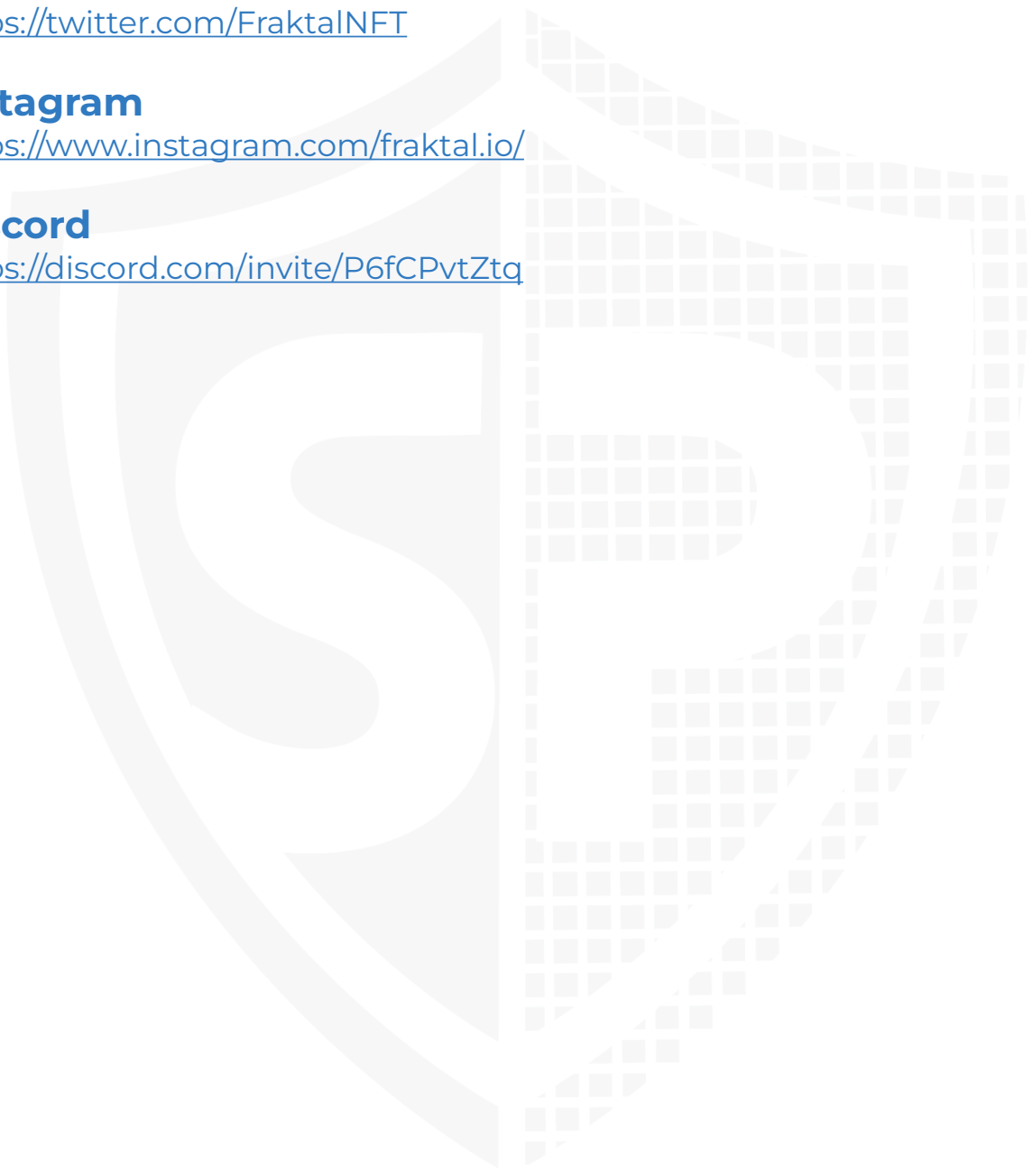
Description

Fraktal is a community first project, with a mission to to empower artists to be in full control of their work and have unlimited creative freedom.

Project Engagement

During the 23rd of February 2022, **Fraktal 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

- Github
    - https://github.com/FraktalNFT/contracts
    - Commit: 99aaeaac28ddfcff49b3ff1d3d9b51969aedb60f

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/access/OwnableUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/proxy/ClonesUpgradeable.sol	3
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	3
@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC1155/ERC1155Upgradeable.sol	3
@openzeppelin/contracts-upgradeable/token/ERC1155/utils/ERC1155HolderUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC721/utils/ERC721HolderUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol	3
@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol	1
@openzeppelin/contracts/proxy/ERC1967/ERC1967Proxy.sol	1
@openzeppelin/contracts/token/ERC1155/IERC1155.sol	1
@openzeppelin/contracts/token/ERC1155/utils/ERC1155Holder.sol	1
@openzeppelin/contracts/token/ERC721/IERC721.sol	1
@openzeppelin/contracts/utils/structs/EnumerableMap.sol	2
@openzeppelin/contracts/utils/structs/EnumerableSet.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/FraktalFactory.sol	bf21f535b72b788b604ce08f18aba445386a6bfd
contracts/IFraktalNFT.sol	579589bb8a28cd6cb5f1276e323ebc2db8f74eb9
contracts/PaymentSplitterUpgradeable.sol	d853821ad97e7fce819dc566aa341ae4326f7cfa
contracts/TransparentUpgradeableProxy.sol	e30ee41beab2ba8951eeef7d21da31f909655cf6
contracts/IPaymentSplitter.sol	9c366ac852cb83dbcc02559da937f7d0fc10c22a
contracts/FraktalNFT.sol	a26e08f227e913463f4b0983c7ff93a94c40d8b6
contracts/FraktalMarket.sol	c46adec5fad7bced31645fa77d30fddd3169c1ff

Metrics

Source Lines
v1.0
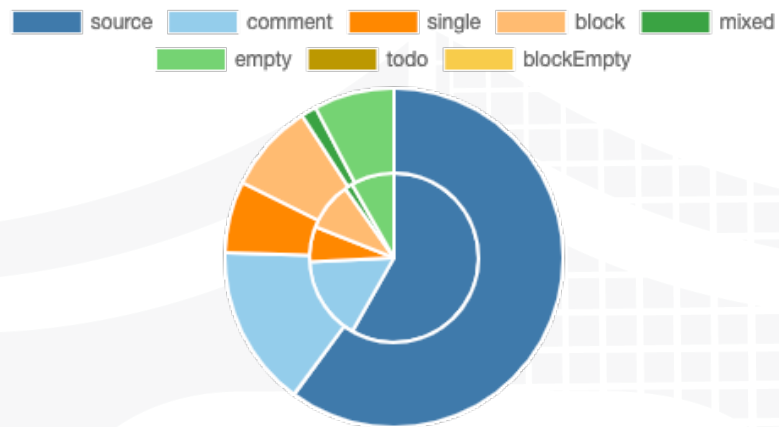
Risk Level
v1.0
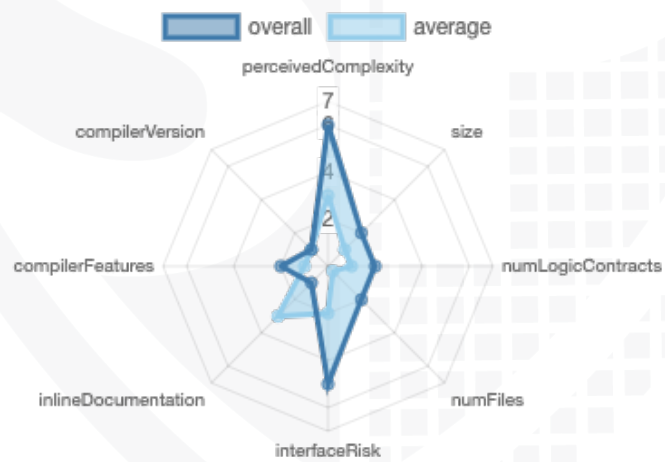
Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	5	0	2	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.

Version	Public	Payable
1.0	85	12

Version	External	Internal	Private	Pure	View
1.0	75	61	1	1	27

State Variables

Version	Total	Public
1.0	41	23

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`^0.8.0`		yes		

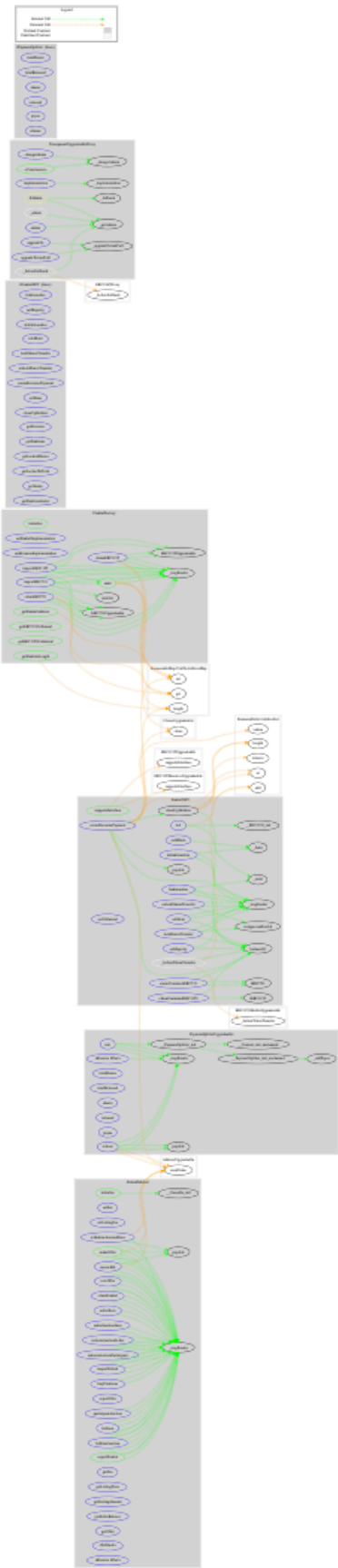| Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	EC Recover	New/ Create/ Create2
1.0				yes		

Inheritance Graph
v1.0

CallGraph
v1.0

Scope of Work/Verify Claims

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

We will verify the following claims:
1. 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)

Correct implementation of Token standard

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 v1.0

FraktalNFT

- claimContainedERC11...
- claimContainedERC721
- createRevenuePayment
- defraktionalize
- fraktionalize
- init
- lockSharesTransfer
- onERC1155BatchRec...
- onERC1155Received
- onERC721Received
- safeBatchTransferFrom
- safeTransferFrom
- sellItem
- setApprovalForAll
- setCollateral
- setMajority
- soldBurn
- unlockSharesTransfer

FraktalFactory

- claimERC1155
- claimERC721
- importERC1155
- importERC721
- initialize
- mint
- onERC1155BatchRec...
- onERC1155Received
- onERC721Received
- renounceOwnership
- setFraktalImplementa...
- setRevenueImplement...
- transferOwnership

FraktalMarket

- buyFraktions
- claimFraktal
- exportFraktal
- importFraktal
- initialize
- listItem
- listItemAuction
- makeOffer
- onERC1155BatchRec...
- onERC1155Received
- participateAuction
- redeemAuctionPartici...
- redeemAuctionSeller
- rejectOffer
- renounceOwnership
- rescueEth
- setFee
- setListingFee
- transferOwnership
- unlistAuctionItem
- unlistItem
- voteOffer
- withdrawAccruedFees

PaymentSplitterUpgradeable

- init
- release

TransparentUpgradeable Proxy

- admin
- changeAdmin
- implementation
- upgradeTo
- upgradeToAndCall

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✗

Comments:

v1.0

• Everybody can mint

Deployer cannot burn or lock user funds

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

Comments:

v1.0

- Everybody can burn

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	–	–	–

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

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

Modifiers and public functions
v1.0

- ⌄ 🔷 initialize
  - ◎ initializer
- ⌄ 🔷 setFraktalImplementation
  - ◎ onlyOwner
- ⌄ 🔷 setRevenueImplementation
  - ◎ onlyOwner
- 🔷 mint
- 🔷 importERC721
- 🔷 importERC1155
- 🔷 claimERC721
- 🔷 claimERC1155

- ⌄ 🔷 init
  - ◎ initializer
- 🔷 fraktionalize
- 🔷 defraktionalize
- 🔷 setMajority
- 🔷 soldBurn
- 🔷 lockSharesTransfer
- 🔷 unlockSharesTransfer
- 🔷 createRevenuePayment 💰
- 🔷 sellItem 💰
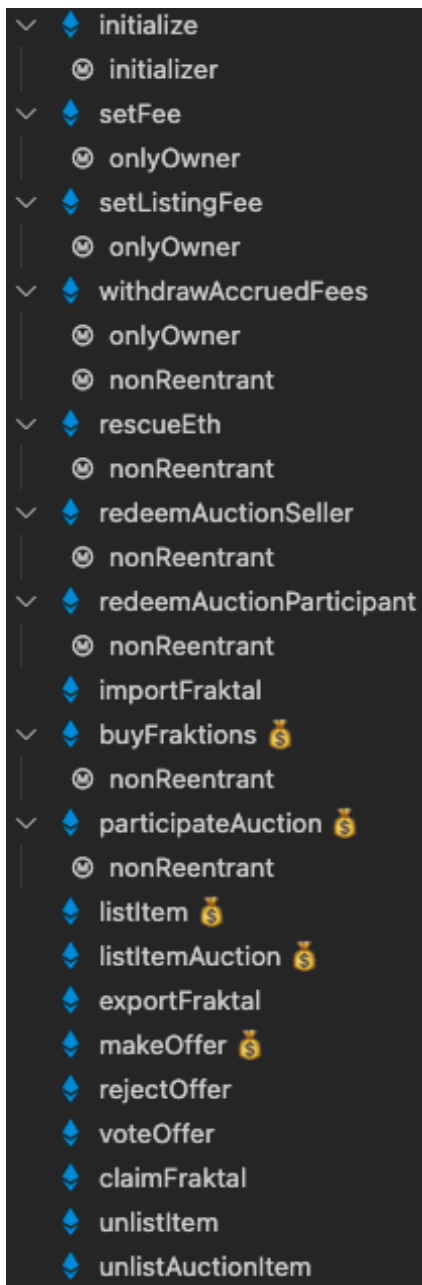- 🔷 claimContainedERC721
- 🔷 claimContainedERC1155
- 🔷 setCollateral

- 🔷 release

- ⌄ 🔷 admin
  - ◎ ifAdmin
- ⌄ 🔷 implementation
  - ◎ ifAdmin
- ⌄ 🔷 changeAdmin
  - ◎ ifAdmin
- ⌄ 🔷 upgradeTo
  - ◎ ifAdmin
- ⌄ 🔷 upgradeToAndCall 💰
  - ◎ ifAdmin

- initialize
  - initializer
- setFee
  - onlyOwner
- setListingFee
  - onlyOwner
- withdrawAccruedFees
  - onlyOwner
  - nonReentrant
- rescueEth
  - nonReentrant
- redeemAuctionSeller
  - nonReentrant
- redeemAuctionParticipant
  - nonReentrant
- importFraktal
- buyFraktions 💰
  - nonReentrant
- participateAuction 💰
  - nonReentrant
- listItem 💰
- listItemAuction 💰
- exportFraktal
- makeOffer 💰
- rejectOffer
- voteOffer
- claimFraktal
- unlistItem
- unlistAuctionItem

Comments

Information: Not listed functions are from imported libraries

- Deployer can set following addresses
  - FraktalFactory
    - FraktalImplementation
    - revenueChannelImplementation

- FraktalMarket
  - Following state variable can be set to 99 percent
    - Fee
    - listingFee
  - Owner can withdraw feesAccrued

- listItem must have more mag.value than listingFee which can be set by the owner. The same for listItemAuction
- collateral can only be set by factory

Please check if an OnlyOwner or similar restrictive modifier has been forgotten.

Source Units in Scope
v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/FraktalFactory.sol	1	————	167	167	140	15	146	————
🔍	contracts/IFraktalNFT.sol	————	1	20	5	3	1	31	————
📝	contracts/PaymentSplitterUpgradeable.sol	1	————	187	178	98	55	81	💰
📝	contracts/TransparentUpgradeableProxy.sol	1	————	123	123	41	72	43	💰🎟
🔍	contracts/IPaymentSplitter.sol	————	1	11	5	3	1	13	————
📝	contracts/FraktalNFT.sol	1	————	293	263	190	58	188	💰🎟
📝	contracts/FraktalMarket.sol	1	————	477	434	370	39	311	💰
📝🔍	**Totals**	**5**	**2**	**1278**	**1175**	**845**	**241**	**813**	💰🎟

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	Fraktal NFT	Everybody can burn	114	Everybody can use soldBurn function L114 to burn any nft from other addresses

Low issues

Issue	File	Type	Line	Description
#1	FraktalFactory	A floating pragma is set	2	The current pragma Solidity directive is „"^0.8.0"".
#2	Fraktal Market	A floating pragma is set	2	The current pragma Solidity directive is „"^0.8.0"".
#3	Fraktal NFT	A floating pragma is set	2	The current pragma Solidity directive is „"^0.8.0"".
#4	PaymentSplitter Upgradeable	A floating pragma is set	2	The current pragma Solidity directive is „"^0.8.0"".
#5	FraktalFactory	Missing Zero Address Validation (missing-zero-check)	43, 50, 54	Check that the address is not zero

#6	Fraktal NFT	Missing Zero Address Validation (missing-zero-check)	58, 281	Check that the address is not zero
#7	Fraktal NFT	State variable visibility is not set	18, 19, 20, 23, 31	It is best practice to set the visibility of state variables explicitly
#8	Fraktal Market	State variable visibility is not set	39	It is best practice to set the visibility of state variables explicitly
#9	PaymentSplitter Upgradeable	State variable visibility is not set	36, 37, 39	It is best practice to set the visibility of state variables explicitly
#10	Fraktal Market	Tautology or contradiction	96, 103	Fix the incorrect comparison by changing the value type or the comparison

Informational issues

Issue	File	Type	Line	Description
#1	Fraktal Market	Require message is missing	All require statements	Provide an error message to the require statement
#2	Fraktal NFT	Require message is missing	All require statements	Provide an error message to the require statement
#3	TransparentUpgradeableProxy	Functions that are not used	112	Remove unused functions

Commented Code exist

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

File	Line	Comment
FraktalFactory	38-41	// constructor(address _implementation, address _revenueChannelImplementation) { // FraktalImplementation = _implementation; // revenueChannelImplementation = _revenueChannelImplementation; // }
	81	// ERC721Upgradeable(_tokenAddress).transferFrom(_msgSender(), address(this), _tokenId);

	95	// ERC1155Upgradeable(_tokenAddress).safeTransferFrom(_msgSender(), address(this), _tokenId, 1, '');
FraktalNFT	54	// constructor() initializer {}

	88-90	// require(this.balanceOf(_msgSender(), 0) == 1);
		// require(fraktionalized == false);
		// require(indexUsed[_tokenId] == false);

	108-109	// require(newValue <= 10000*10**18); // require(newValue > 0);
	119-121	// if (_msgSender() != owner) { // require(isApprovedForAll(owner, _msgSender())); // }
	184	// uint256 bal = this.balanceOf(holders.at(i), fraktionsIndex);

	224-246	// function getRevenue(uint256 index) external view returns (address) {
		// return revenues.get(index);
		// }
		// function getFraktions(address who) external view returns (uint256) {
		// return this.balanceOf(who, fraktionsIndex);
		// }
		// function getLockedShares(uint256 index, address who)
		// external
		// view
		// returns (uint256)
		// {
		// return lockedShares[index][who];
		// }
		// function getLockedToTotal(uint256 index, address who)
		// external
		// view
		// returns (uint256)
		// {
		// return lockedToTotal[index][who];
		// }

	258-259	// require(fraktionalized==false); // require(IERC721(contractAddress).ownerOf(index) == address(this));

	286-292	// function getStatus() external view returns (bool) {
		// return sold;
		// }
		// function getFraktionsIndex() external view returns (uint256) {
		// return fraktionsIndex;
		// }

Payment SplitterUpgradeable	138	// uint256 bal = IFraktalNFT(tokenParent).getFraktions(_msgSender());

Recommendation
Remove the commented code, or address them properly.

Audit Comments
25. February 2022:
· 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	**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**

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY