SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Avanzo

Audit

Security Assessment

24.August,2022

For

avanzo

EXPANDING CRYPTO ESTATES

Disclaimer 2
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
Inheritance Graph 12
CallGraph 13
Scope of Work/Verify Claims 14
Modifiers and public functions 24
Source Units in Scope 25
Critical issues 26
High issues 26
Medium issues 26
Low issues 26
Informational issues 27
Audit Comments 27
SWC Attacks 28

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	21.August,2022	• Layout project • Automated- /Manual-Security Testing • Summary

Network

Binance (BSC)

Website
https://avanzo.io/

Twitter
https://twitter.com/Avanzocrypto

Telegram
@AvanzoCrypto

YouTube
https://youtu.be/YP033Zy6CdA

Description

In the event that readers choose to participate in one of our investing pools, a new investing strategy is proposed in this article. By tying investors' investments to real estate assets, which equal the invested value for each pool, this approach will lower the risk that investors face while dealing with the volatility of the cryptocurrency market. With a return provided to investors on various cryptocurrencies according to their preferences, utilizing a DAO vote, investors can rest certain that their investments are safe and the danger of capital loss is eliminated. Using a DAO vote, investors also have the option to (end) or (continue) the investment after receiving each return.

Project Engagement

During the 21th of August 2022, **Avanzo** 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

Contract Links

v1.0

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.

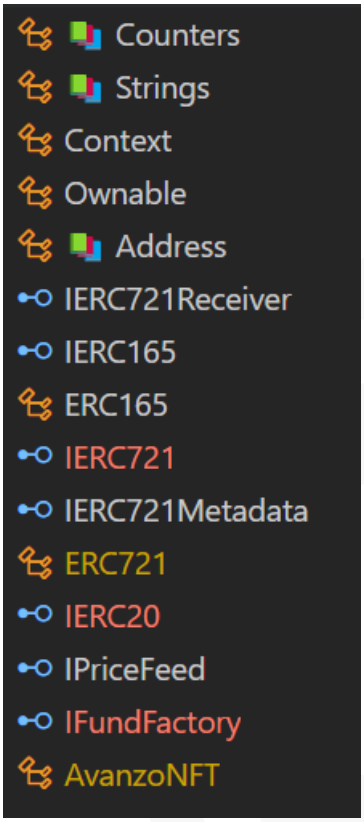
2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analyzing a program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

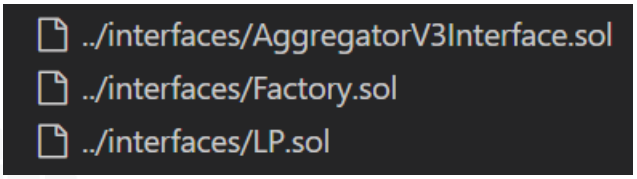
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

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:





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.

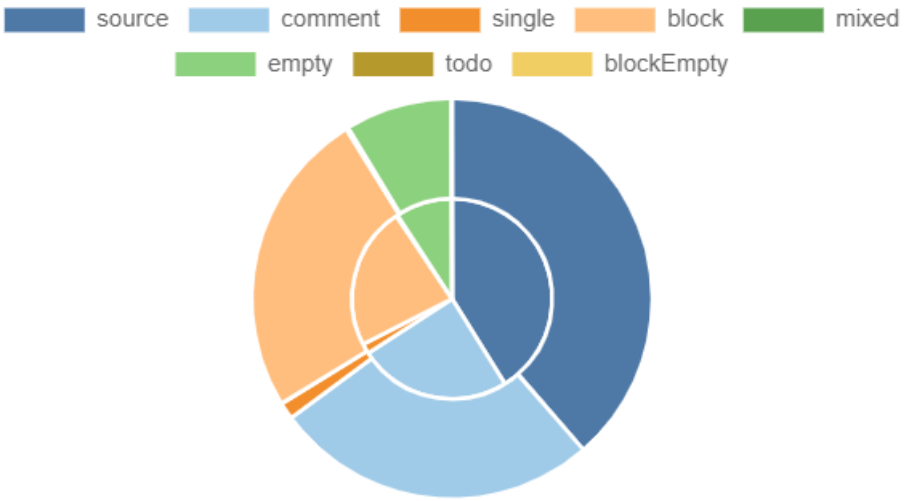
v1.0

File Name	SHA-1 Hash
contracts/interfaces/AggregatorV3Interface.sol	16959df5f11a22380937da70c3f24d6e5e0f5027
contracts/interfaces/Factory.sol	671bc43f43ed80b214dedaa2d02d8fee92cbaa82
contracts/interfaces/LP.sol	4dd693ce8cca59e1b40e8105efac105e91ab6f75
contracts/PriceFeed.sol	3758b4e581ca68c9bb8074de58e3d679095a4afb
contracts/Marketplace.sol	36e24b583229303c4a5f8c6105bc027ae61acbe3
contracts/FundFactory.sol	860d6a32f4dab6651ba02f56af4149c916784bf9
contracts/NFT_flat.sol	2598febc5c494ae52999a331317f27d983029bc2

Metrics

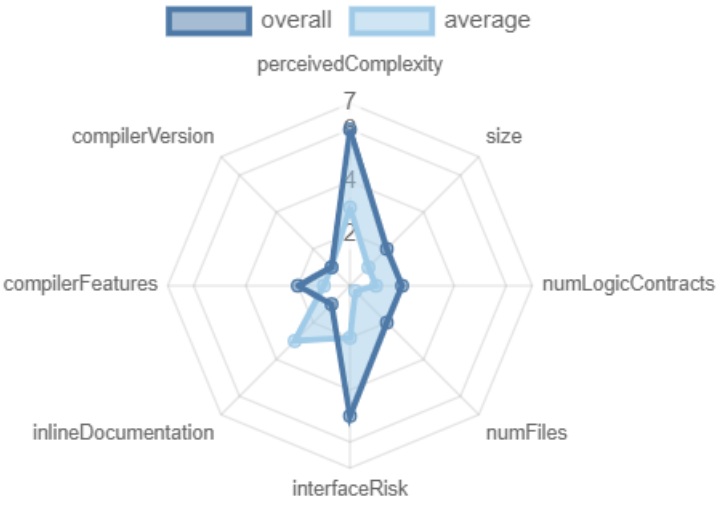
Source Lines

v1.0



Risk Level

v1.0



Capabilities

v1.0

Components

Contracts	Libraries	Interfaces	Abstract
5	3	13	3

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
112	0

External	Internal	Private	Pure	View
55	125	8	6	71

StateVariables

Total	Public
47	8

Capabilities

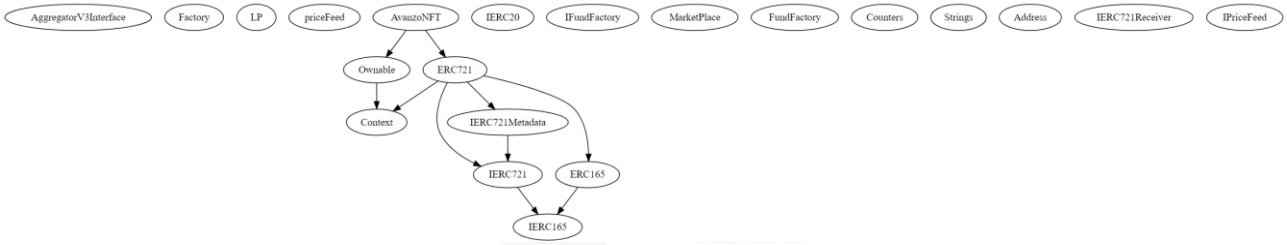
Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
`^0.8.2` `^0.8.4`			`yes` (2 asm blocks)	

Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/Create/Create2
`yes`		`yes`			`yes` → `NewContract:AvanzoNFT`

TryCatch	Σ Unchecked
`yes`	`yes`

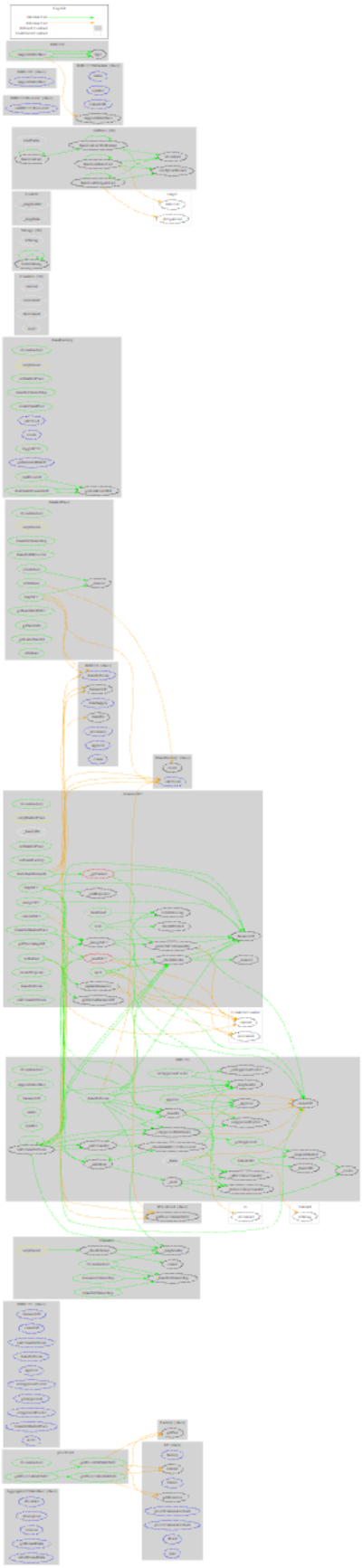
Inheritance Graph

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			

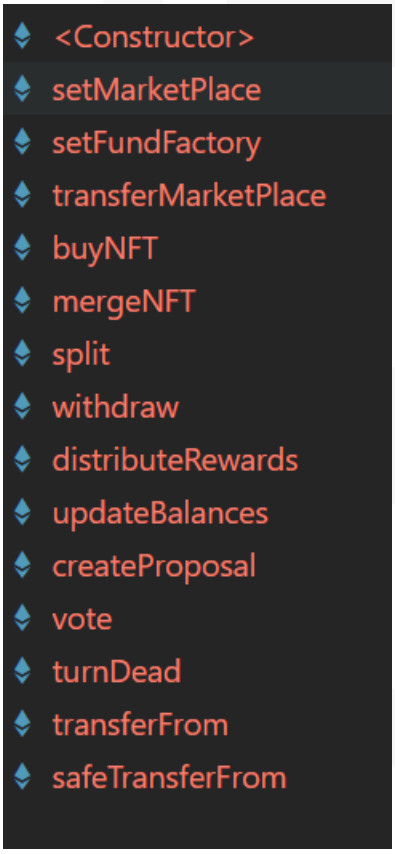
ERC721				
Function	**Description**	**Exist**	**Tested**	**Verified**
BalanceOf	Count all NFTs assigned to an owner			
OwnerOf	Find the owner of an NFT			
SafeTransferFrom	Transfers the ownership of an NFT from one address to another address			
SafeTransferFrom	See above - Difference is that this function has an extra data parameter			
TransferFrom	Transfer ownership of an NFT			
Approve	Change or reaffirm the approved address for an NFT			
SetApprovalForAll	Enable or disable approval for a third party ("operator") to manage all of `msg.sender`'s assets			
GetApproved	Get the approved address for a single NFT			
IsApprovedForAll	Query if an address is an authorized operator for another address			

SupportsInterface	Query if a contract implements an interface			
Name	Provides information about the name			
Symbol	Provides information about the symbol			
TokenURI	Provides information about the TokenUri			

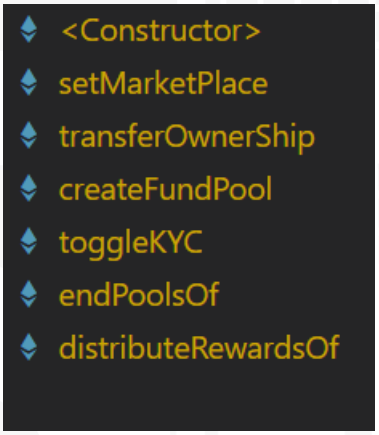
Write functions of contracts

v1.0

NFT_Flat



FundFactory



Marketplace

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint			
Max / Total Supply	N/A		

Comments:

- The buyers can mint/buy their NFTs if their KYC is approved by the owner and they invest a minimum of $500.

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			

Comments

Only the splitting of NFTs will be paused when there will be a voting going on.

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 owner can change the royalty address to any address of their choosing and all the royalties will be transferred to that wallet on every successful trade of the NFTs.

Deployer cannot blacklist/antisnipe addresses

Name	Exist	Tested	Status
Deployer can blacklist/antisnipe addresses			

Overall checkup (Smart Contract Security)

Tested	Verified

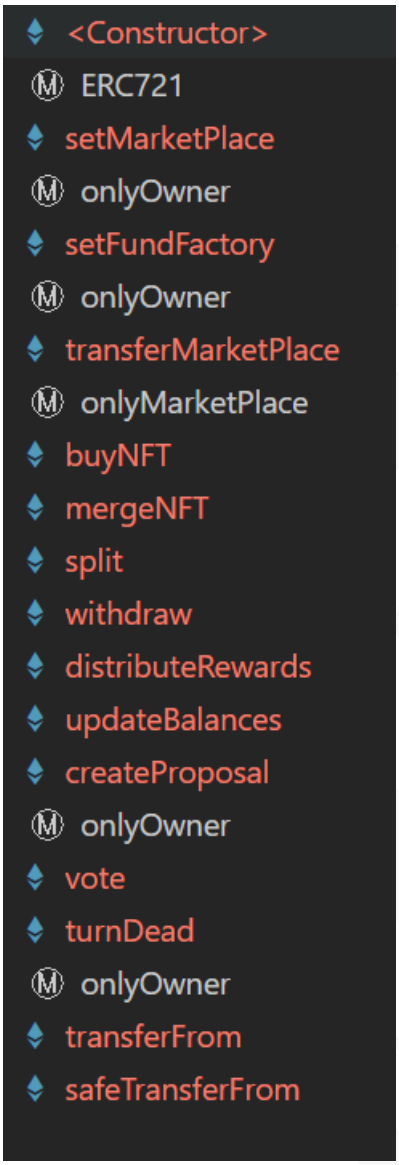
Legend

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

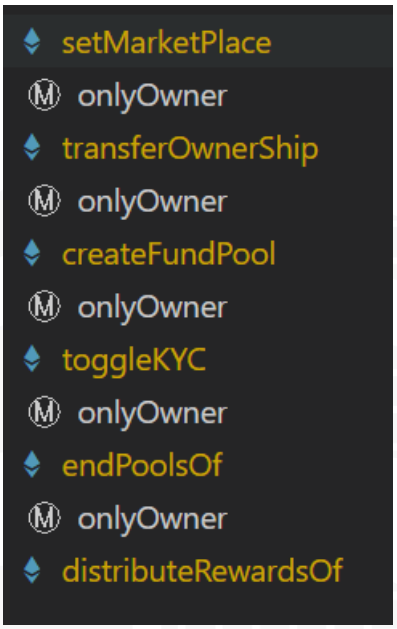
Modifiers and public functions

v1.0

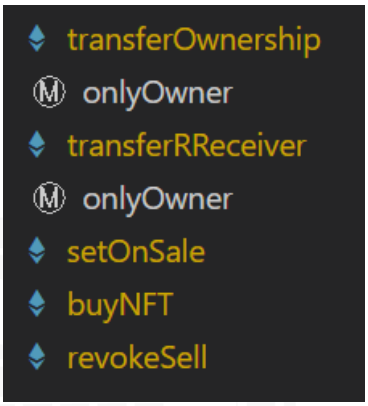
NFT_Flat



FundFactory



Marketplace



Comments:

- The owner can start and stop(only when the voting is stopped) fund pools
- The owner can create a proposal for VOTE
- The owner can set a new marketplace and FundFactory address
- The owner can transfer NFT without approval to any arbitrary address of their choosing
- The owner can set the KYC status for the users

Source Units in Scope

v1.0

File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/interfaces/AggregatorV3Interface.sol		1	11	6	3	1	11
contracts/interfaces/Factory.sol		1	7	6	3	1	3
contracts/interfaces/LP.sol		1	14	6	3	1	17
contracts/PriceFeed.sol	1		58	58	34	17	26
contracts/Marketplace.sol	1	3	206	168	110	34	119
contracts/FundFactory.sol	1		134	134	85	28	108
contracts/NFT_flat.sol	8	7	1602	1352	714	646	620
Totals	**11**	**13**	**2032**	**1730**	**952**	**728**	**904**

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	NFT_Flat.sol	Missing Events	1273,1280,1288,1326,1356	Emit an event for critical parameter changes.
#2	NFT_Flat.sol	Missing zero check	1158,1273	Check that the address is not zero
#3	NFT_Flat.sol	Missing error message	1398,1405	Always add error messages in require checks to avoid any confusion
#4	FundFactory.sol	Missing zero check	17,37,44	Check that the address is not zero
#5	FundFactory.sol	Missing Events	37,44,52,85	Emit an event for critical parameter changes.
#6	Marketplace.sol	Missing zero check	55,71,78	Check that the address is not zero
#7	Marketplace.sol	Missing Events	71,78,85,99,124,	Emit an event for critical parameter changes.
#8	All	Floating Pragma	-	The current pragma Solidity directives are ‘^0.8.4’ and ‘0.8.2’. Contracts should be deployed

				with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using other versions.

Informational issues

Issue	File	Type	Line	Description
#1	FundFactory.sol	Unused return value	125	Make sure that all the return values are checked
#2	FundFactory.sol	Confusing error messages	126,127,128	Error messages should be more informative
#3	NFT_Flat.sol	Uninitialized Local Variable	1247,1414	Make sure that all the variables are initialized.

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.

23. August, 2022:

- There is still an owner (Owner still has not renounced ownership)
- Some comments in the code doesn’t match the actual code which will confuse developers/investors. NFT_Flat.sol – L1182
- The contracts contain and use the code from the interfaces. The actual code of some of these interfaces was not provided to us and the function can do anything.
- Read the whole report and modifiers section 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-	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	
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	
SWC-	Write to Arbitrary	CWE-123: Write-what-where Condition	

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	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