



SOLIDProof
Bring trust into your projects

**Blockchain Security | Smart Contract Audits | KYC
Development | Marketing**

MADE IN GERMANY

Virgo Audit

**Security Assessment
17. March, 2023**

For



SolidProof_io



@solidproof_io

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	11
Risk Level	11
Capabilities	12
Inheritance Graph	13
CallGraph	14
Scope of Work/Verify Claims	15
Modifiers and public functions	25
Source Units in Scope	29
Critical issues	30
High issues	30
Medium issues	30
Low issues	30
Informational issues	30
Commented Code exist	31
Audit Comments	31
SWC Attacks	33

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 analyse.

Version	Date	Description
1.0	10 March 2023 to 15. March 2023	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary

Network

Ethereum

Website

<https://messier.app/>

Telegram

<https://t.me/MessierM87Community>

Twitter

<https://twitter.com/MessierM87>

Medium

<https://github.com/MessierM87>



Description

Our main goal is to create decentralized applications that will provide both consumers and businesses with tools designed to make cryptocurrency transactions more confidential, secure, and viable than conventional currencies.

The dapps that Messier creates, are aimed at the general public, but their specific use cases may vary. Generally, the main entities that will want to use our dapps are individuals or organizations looking to utilize decentralized applications for a variety of purposes within the field of financial transactions.

Project Engagement

During the Date of 09 March 2023, **Messier Team** engaged Solidproof.io to audit smart contracts that they created for the **Virgo DAPP**. 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 white paper.

Logo



Contract Link

v1.0

- https://github.com/MessierM87/VIRGO_SolidityContracts
- Commit: d98832a

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:

Dependency / Import Path	Count
@openzeppelin/contracts-upgradeable/access/IAccessControlEnumerableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol	2
@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/CountersUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol	1
@openzeppelin/contracts/access/AccessControlEnumerable.sol	1
@openzeppelin/contracts/access/Ownable.sol	4
@openzeppelin/contracts/token/ERC20/ERC20.sol	5
@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol	1
@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol	1
@openzeppelin/contracts/utils/Context.sol	1
@openzeppelin/contracts/utils/math/SafeMath.sol	1
@uniswap/v2-core/contracts/interfaces/IUniswapV2Factory.sol	1
@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol	1
@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router01.sol	1
@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol	2

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

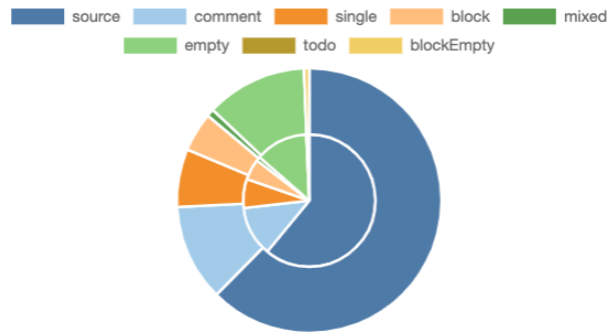
File Name	SHA-1 Hash
contracts/MarketPlace.sol	ca9645b3c61964ba9fdc8effe33d949c12366aff
contracts/libs/M87Bank.sol	6725300982978df01e0377b5db62d78c46c432cf
contracts/libs/xToken.sol	ebd81a8a8c56697b06943faf1d1d9a65cc95202b
contracts/libs/IUniSwap.sol	c48281fe667a22a0dad3699022f29c65390dd69
contracts/libs/MESSIER.sol	b3bf8f3a3e0ddc0dceae1ddb54441f4720345611
contracts/libs/MOTTToken.sol	92f4bb1e4e0f72bac2791f8839dec2d1b55f7d6b
contracts/libs/IMessierNFT.sol	5032fef30a77caa6f2f8af75a0d81b075466a452
contracts/libs/simulatorM87.sol	3d448ced3ae3cb98595904b68700850dd1ddf7c7
contracts/libs/MTTToken.sol	11c732dd5fc459d32365685260edfefe0d435365
contracts/libs/MessierNFT.sol	a80083baf52800020307572ff57108f7f773e9a0
contracts/libs/Pausable.sol	b94e4418d6caa6b2cc601ea21faee00134deed01

contracts/libs/IM87.sol	9247fbca437a2aa4826abc884a56153f6551f3fd
contracts/libs/ ISupernova.sol	3e1582c7baee6e4210d00295c4c53a50e9f9c1f8
contracts/Supernova.sol	8b3628dec116c114eea5b6db337e214209b58e21
contracts/Dao.sol	f2dd90cc440304545314c334f67898474f9097d6

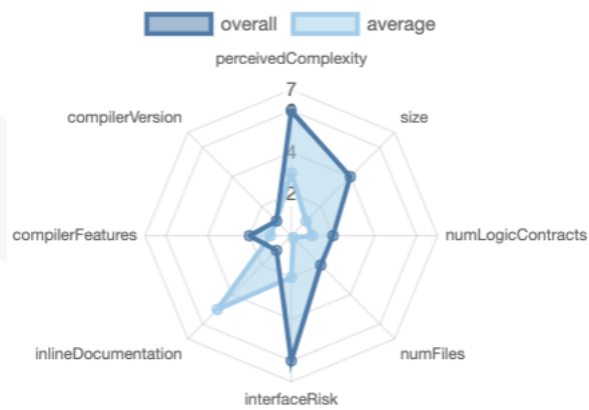


Metrics

Source Lines v1.0



Risk Level v1.0



Capabilities

Components

 Contracts	 Libraries	 Interfaces	 Abstract
13	0	4	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.






 Public	 Payable
161	15







External	Internal	Private	Pure	View
81	177	15	0	61

StateVariables

Total	 Public
137	39

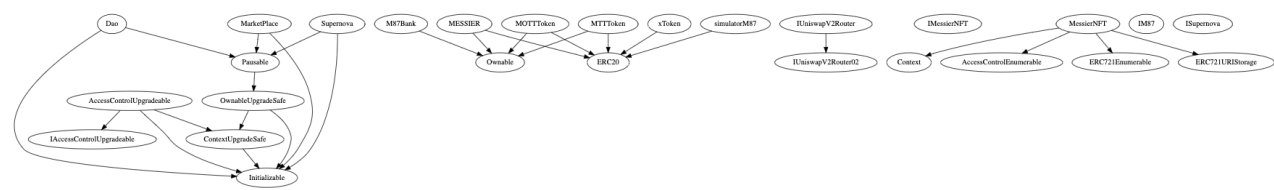
Capabilities

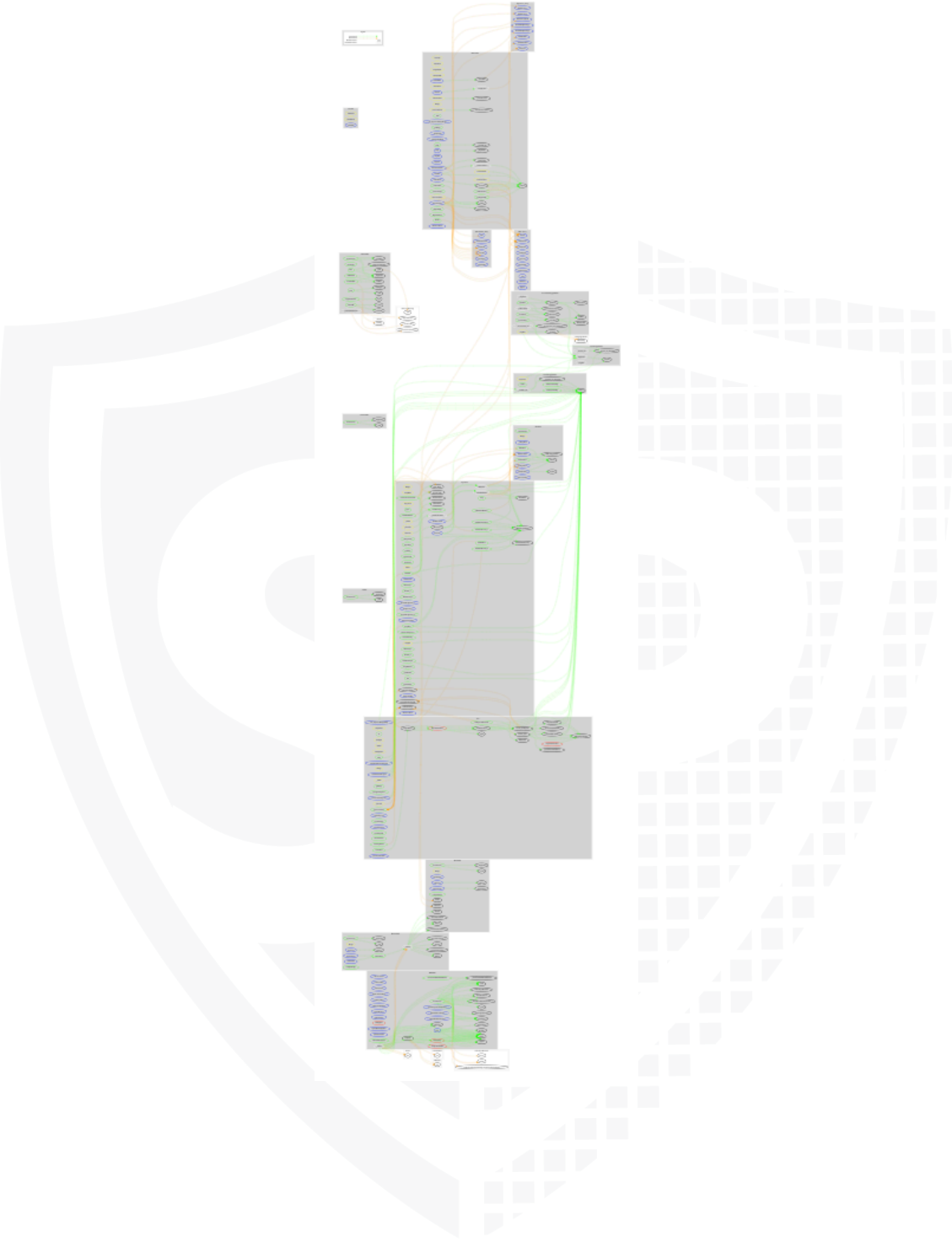
Solidity Versions observed	 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
 0.8.9		yes		

 Transfers ETH	 Low-Level Calls	 DelegateCall	 Uses Hash Functions	 ECRrecover	 New/Create/Create2
yes			yes		yes → NewContract:M87Bank

Inheritance 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 cannot set fees
7. Deployer cannot blacklist/antisnipe addresses
8. Overall checkup (Smart Contract Security)



Is contract an upgradeable

Name	
Are any contracts upgradeable?	Yes

Comments:

v1.0

- Owner can deploy a new version of the **Supernova and Pausable** contracts which can change any limit and give owner new privileges
 - Be aware of this and do your own research for the contract which is the contract pointing to

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	✓	✓	✓

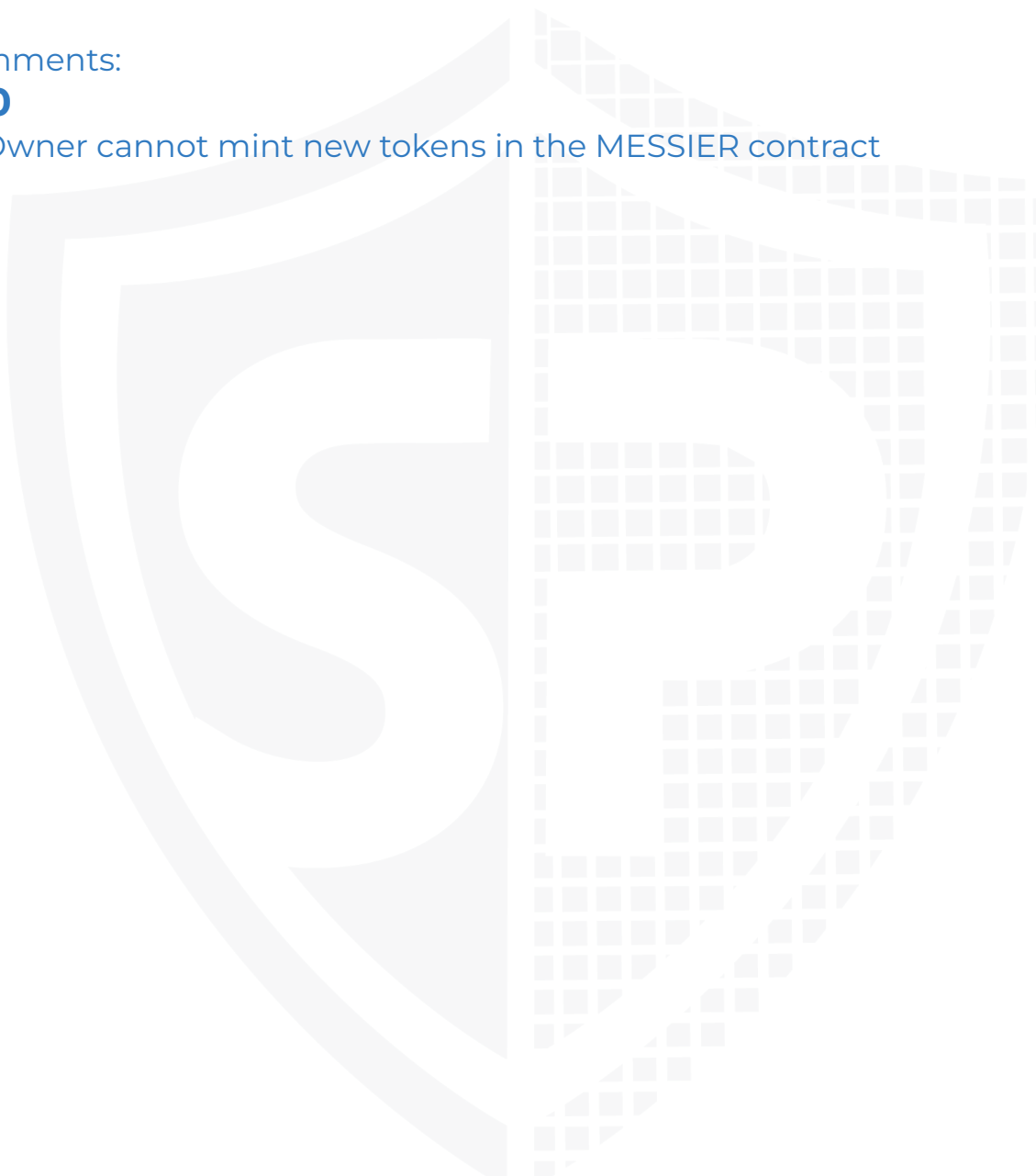
Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✓
Max / Total Supply	1.000.000.000.000		

Comments:

v1.0

- Owner cannot mint new tokens in the MESSIER contract



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:

v1.0

- Owner can pause contract



Deployer cannot set fees

Name	Exist	Tested	Status
Deployer cannot set fees over 25%	✓	✓	✓
Deployer cannot set fees to nearly 100% or to 100%	✓	✓	✓

Comments:

v1.0

- Fees cannot be set without any limitations in the 'MESSIER.sol' contract

Deployer can blacklist/antisnipe addresses

Name	Exist	Tested	Status
Deployer can blacklist/antisnipe addresses	✓	✓	✗

Comments:

v1.0

- Owner is able to blacklist addresses in the 'MESSEIR.sol' contract



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

M87Bank.sol

- ◆ _ChangeHash
- ◆ WithdrawEth 💰
- ◆ WithdrawToken 💰

MESSIER.sol


- ◆ enableTrading
- Ⓜ onlyOwner
- ◆ removeLimits
- Ⓜ onlyOwner
- ◆ disableTransferDelay
- Ⓜ onlyOwner
- ◆ setEarlySellTax
- Ⓜ onlyOwner
- ◆ updateSwapTokensAtAmount
- Ⓜ onlyOwner
- ◆ updateMaxTxnAmount
- Ⓜ onlyOwner
- ◆ updateMaxWalletAmount
- Ⓜ onlyOwner
- ◆ excludeFromMaxTransaction
- Ⓜ onlyOwner
- ◆ updateSwapEnabled
- Ⓜ onlyOwner
- ◆ updateBuyFees
- Ⓜ onlyOwner
- ◆ updateSellFees
- Ⓜ onlyOwner
- ◆ excludeFromFees
- Ⓜ onlyOwner
- ◆ setAutomatedMarketMakerPair
- Ⓜ onlyOwner
- ◆ updateMarketingWallet
- Ⓜ onlyOwner
- ◆ updateDevWallet
- Ⓜ onlyOwner
- ◆ Send
- Ⓜ onlyOwner

DAO.sol

- ◆ setup
- Ⓜ initializer
- ◆ _Set_SupernovaBridgeBridge
- Ⓜ onlyOwner
- ◆ createCashOutProposal
- Ⓜ _IsPowehi
- ◆ createDoubleClassicProposal
- Ⓜ _IsPowehi
- ◆ createStructureProposal
- Ⓜ _IsPowehi
- ◆ signOnProposal
- Ⓜ _IsPowehi
- Ⓜ notSigner
- Ⓜ atState
- ◆ VoteOnProposal
- Ⓜ _IsHalo
- Ⓜ stateQuorum
- Ⓜ atState
- ◆ CloseProposal
- Ⓜ atState
- Ⓜ OnlyOracle
- ◆ OracleNext
- Ⓜ OnlyOracle
- ◆ FullCycleToExecution

MarketPlace.sol

- ◆ setup
- Ⓜ initializer
- ◆ _Set_SupernovaBridgeBridge
- Ⓜ onlyOwner
- ◆ _ChangeHash
- Ⓜ onlyOwner
- ◆ _ChangeStateManul
- Ⓜ onlyOwner
- ◆ makeBidInit
- Ⓜ correctId
- Ⓜ auctionInit
- Ⓜ minimumBid
- Ⓜ balanceOfToken
- ◆ RefundsEth
- ◆ DropEth
- ◆ Drop
- Ⓜ correctId
- Ⓜ auctionInit
- Ⓜ dropOfToken
- ◆ DetermineWinInit
- ◆ Refunds
- ◆ Resell
- ◆ makeBid 💰
- ◆ AddFundEth 💰
- ◆ AddFundToken 💰
- Ⓜ balanceOfToken
- ◆ DetermineWin
- ◆ <Constructor> 💰



- ◆ setup
- Ⓜ initializer
- ◆ changeOracle
- Ⓜ onlyOwner
- ◆ InjectHalo
- Ⓜ OnlyOracle
- ◆ InjectPowehi
- Ⓜ OnlyOracle
- ◆ returnContractCurrentId
- ◆ _WithdrawToken 💰
- ◆ WithdrawEth 💰
- Ⓜ Bridge
- ◆ PutInReward_1
- Ⓜ Bridge
- ◆ PutInReward_2
- Ⓜ Bridge
- ◆ PutAndDropReward_1
- Ⓜ Bridge
- ◆ DropReward_1
- Ⓜ Bridge
- ◆ PutAndDropReward_2
- Ⓜ Bridge
- ◆ PutInTreasuryETH 💰
- Ⓜ Bridge
- ◆ PutInTreasuryet 💰
- ◆ PutInTreasuryToken
- ◆ PutOutTreasury
- Ⓜ Bridge
- ◆ PutOutTokenTreasuryB
- Ⓜ Bridge
- ◆ StakM87
- ◆ PutDarkList
- ◆ DropDarkList
- ◆ DropM87
- ◆ WithdrawReward_1
- ◆ WithdrawReward_2
- ◆ Received
- Ⓜ Bridge

Ownership/Authority Privileges

S.No	File	Privileges
#1	MESSIER.sol	<ul style="list-style-type: none"> • Enable trading and transfer delay but cannot disable it • Remove limits and cannot add them again • Enable/Disable early sell tax • Update the amount to swap tokens, max transaction amount, and max wallet amount within a safe range • Set fees but it cannot be more than 25% • Include/Exclude wallets from max transaction amount, and fees • Update marketing, and dev wallet address • Set AMM pair address
#2	MOTTToken.sol	<ul style="list-style-type: none"> • Users can only transfer tokens if they pass the right hash value in the transfer function. • The Hash value will be constant after deployment.
#3	M87Bank.sol	<ul style="list-style-type: none"> • The owner is able to set/change the hash in the contract at any time • The users who have access to the hash can withdraw any type of tokens, if available in the contract balance.
#4	DAO.sol (isPowehi addresses, OnlyOracle and owner)	<ul style="list-style-type: none"> • Set the address of Supernova bridge • Create cash out proposal • Create Structure proposal • Only _IsPowehi addresses can sign on proposal. • Only _isHalo addresses can vote on proposals • Only Oracle address can close and checkOut a proposal • Only Oracle address can execute cycles
#5	Marketplace.sol	<ul style="list-style-type: none"> • Set the address of Supernova bridge • Change HASH value, and state

S.No	File	Privileges
#6	Supernova.sol (Owner, onlyOracle)	<ul style="list-style-type: none"> • Change oracle address • Oracle address can Inject halo addresses (and only these addresses can vote in the Dao) , and Power addresses • Users with access to the HASH can: <ul style="list-style-type: none"> - Withdraw ETH, and token - Put in rewards - Drop rewards - Put out treasury

- There are several authorities which are authorized to call some functions, that means, if the owner is renounced, another address is still authorized to call functions
 - Be aware of this

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

Source Units in Scope

v1.0

File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/MarketPlace.sol	1	————	579	533	427	40	227
contracts/libs/M87Bank.sol	1	————	57	57	51	1	48
contracts/libs/xToken.sol	1	————	14	14	8	1	6
contracts/libs/IUniSwap.sol	————	1	9	9	6	1	3
contracts/libs/MESSIER.sol	1	————	498	491	288	125	249
contracts/libs/MOTTToken.sol	1	————	90	67	50	3	41
contracts/libs/IMessierNFT.sol	————	1	18	6	3	1	13
contracts/libs/simulatorM87.sol	1	————	14	14	8	1	6
contracts/libs/MTTToken.sol	1	————	87	64	43	16	36
contracts/libs/MessierNFT.sol	1	————	123	96	68	14	59
contracts/libs/Pausable.sol	4	————	295	295	153	102	106
contracts/libs/IM87.sol	————	1	31	10	5	1	21
contracts/libs/ISupernova.sol	————	1	15	6	3	1	23
contracts/Supernova.sol	1	————	731	718	547	80	356
contracts/Dao.sol	1	————	942	859	679	101	368
Totals	14	4	3503	3239	2339	488	1562

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalised lines of the source unit (e.g. normalises functions spanning multiple lines)
nSLOC	normalised 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

Critical issues

No critical issues

High issues

Issue	File	Type	Line	Description
#1	Messier NFT.sol	Access Control	55	Any arbitrary user can mint NFTs without paying anything until the max supply is reached.

Medium issues

No medium issues

Low issues

Issue	File	Type	Line	Description
#1	All	A floating pragma is set	—	The current pragma Solidity directive is „^0.8.9“.
#2	Dao.sol	Missing Zero Address Validation (missing-zero-check)	203, 239	Check that the address is not zero
#3	Supernova.sol	Missing Zero Address Validation (missing-zero-check)	190, 195, 209	Check that the address is not zero
#4	MESSIER.sol	Missing Zero Address Validation (missing-zero-check)	210, 238, 247, 259, 264	Check that the address is not zero
#5	Supernova.sol	Missing Events Arithmetic	190, 195, 209	Emit an event for critical parameter changes
#6	MTTToken.sol	Redundant Function	59, 66	The “transferNFT” will not work because the call to the transferFrom function is commented.

Informational issues

Issue	File	Type	Line	Description
#1	MOTTTo ken.sol	Misspelling	See description	Change following words: - approveal Make sure to change it everywhere else as well.
#2	Dao.sol	Wrong Error Message	209	The error message displayed doesn't correspond to the condition inside the 'require' statement
#3	All	NatSpec documentation missing	—	If you started to comment your code, also comment all other functions, variables etc.
#5				

Commented Code exist

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

File	Line	Comment
MessierN FT.sol	66	<pre>// function tokenURI(uint256 tokenId) // public // view // virtual // override(ERC721URIStorage, ERC721) // returns (string memory) // { // return super.tokenURI(tokenId); // }</pre>

Recommendation

Remove the commented code, or address them properly.

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information <https://docs.soliditylang.org/en/latest/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. March 2023:

- There is still an owner (Owner still has not renounced ownership)
- Owner can deploy a new version of the contract which can change any limit and give owner new privileges
- Read whole report and modifiers section for more information



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	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	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	NOT 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

*Solid
Proofed*

**Blockchain Security | Smart Contract Audits | KYC
Development | Marketing**


MADE IN GERMANY