SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Jamonswap

Audit

Security Assessment

27. January, 2022

For

JAMONSWAP

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	26. January 2022	• Layout project • Automated- /Manual-Security Testing • Summary
	27. January 2022	Finished

Network

Polygon

Website

https://jamonswap.finance/

Telegram

https://t.me/+4TWEeg5uQX02YTlk
https://t.me/jamonswap

Twitter

https://twitter.com/JamonSwap

Instagram

https://www.instagram.com/JamonSwap/

Description

TBA

Project Engagement

During the 24th of January 2022, **Jamonswap 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/jamonswap/contracts
    - Commit: c9a769356633bfeb8b921cae33b3b19059af74a8

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.

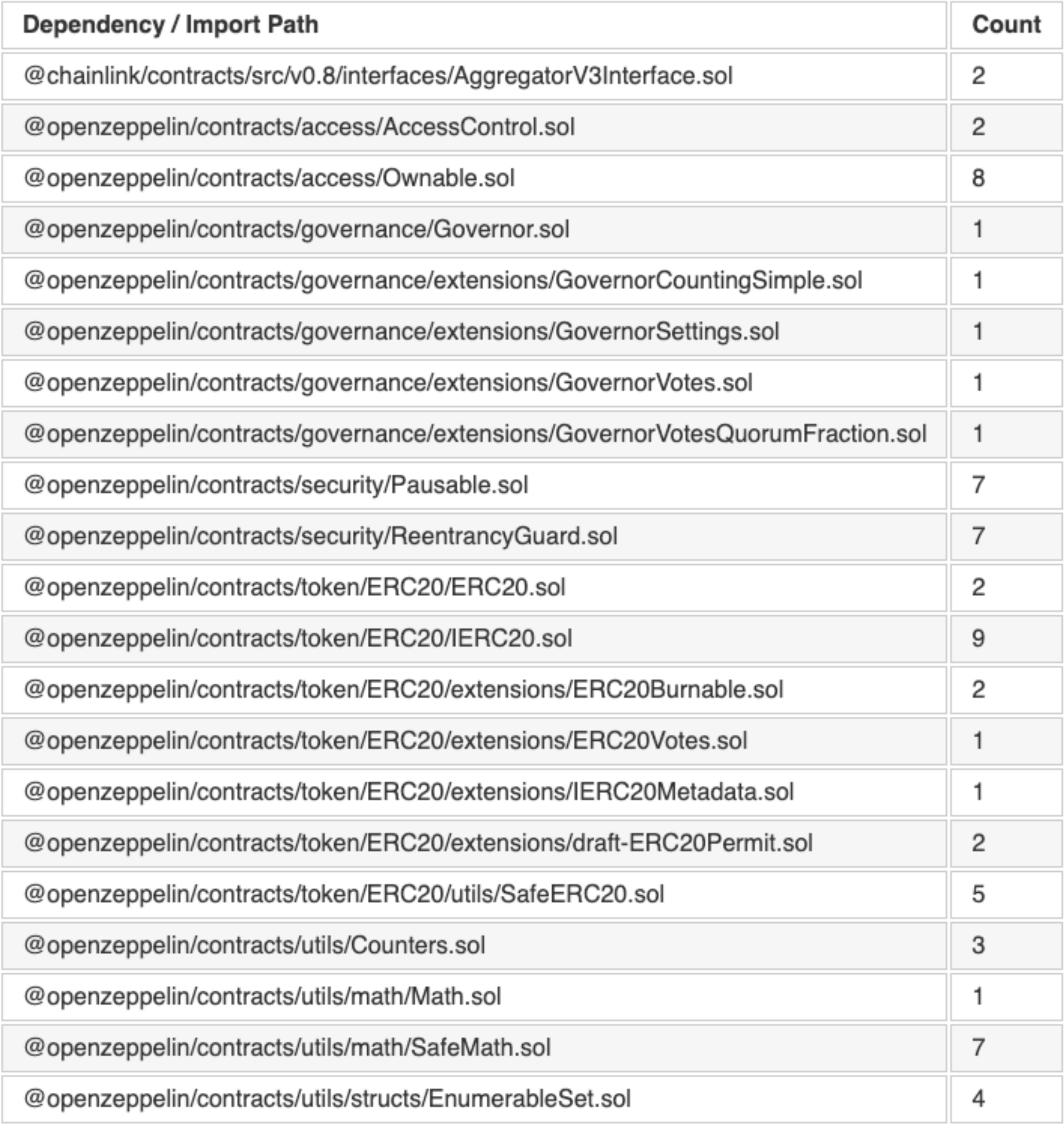
2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analysing a program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



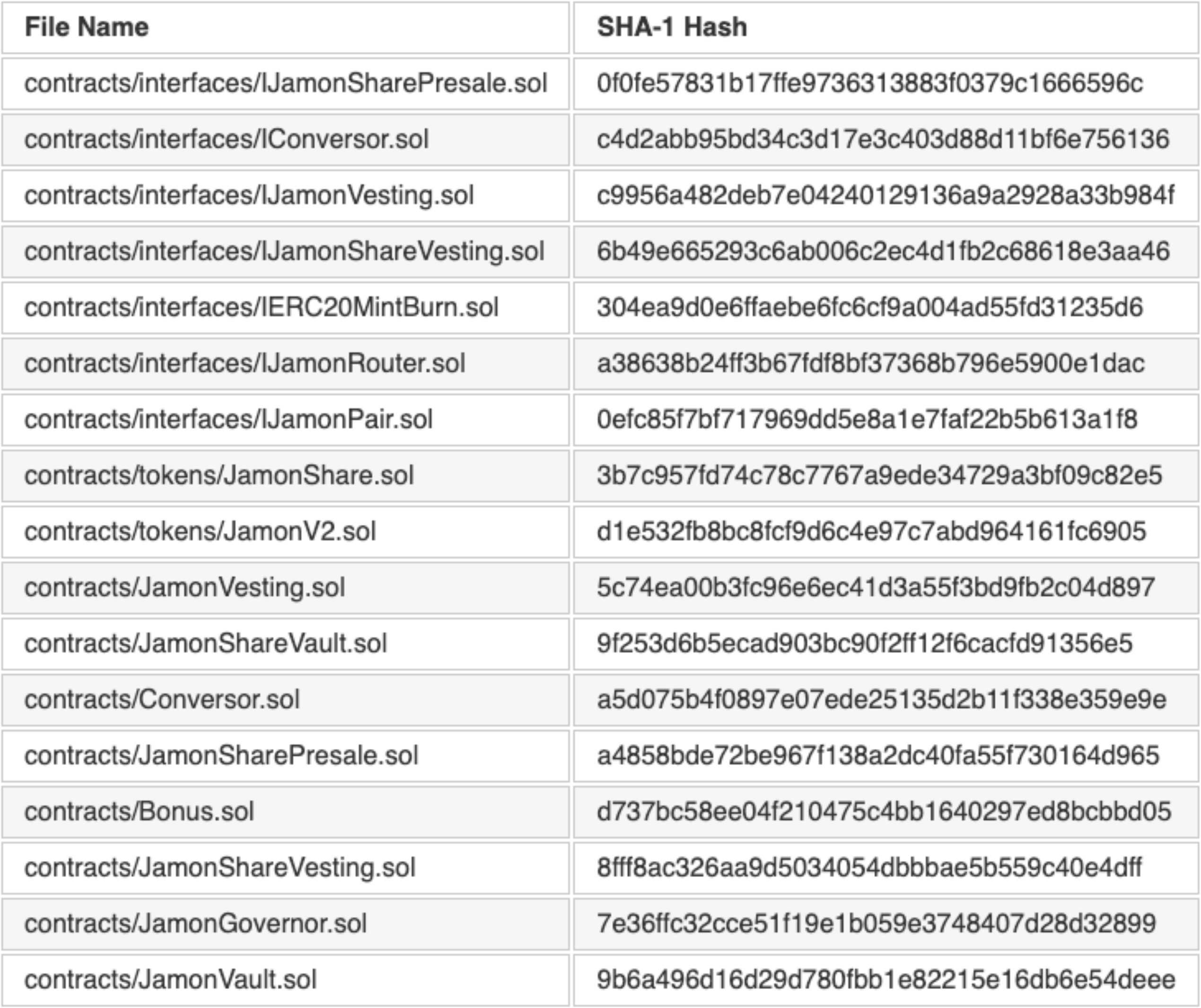
Dependency / Import Path	Count
@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol	2
@openzeppelin/contracts/access/AccessControl.sol	2
@openzeppelin/contracts/access/Ownable.sol	8
@openzeppelin/contracts/governance/Governor.sol	1
@openzeppelin/contracts/governance/extensions/GovernorCountingSimple.sol	1
@openzeppelin/contracts/governance/extensions/GovernorSettings.sol	1
@openzeppelin/contracts/governance/extensions/GovernorVotes.sol	1
@openzeppelin/contracts/governance/extensions/GovernorVotesQuorumFraction.sol	1
@openzeppelin/contracts/security/Pausable.sol	7
@openzeppelin/contracts/security/ReentrancyGuard.sol	7
@openzeppelin/contracts/token/ERC20/ERC20.sol	2
@openzeppelin/contracts/token/ERC20/IERC20.sol	9
@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol	2
@openzeppelin/contracts/token/ERC20/extensions/ERC20Votes.sol	1
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	1
@openzeppelin/contracts/token/ERC20/extensions/draft-ERC20Permit.sol	2
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	5
@openzeppelin/contracts/utils/Counters.sol	3
@openzeppelin/contracts/utils/math/Math.sol	1
@openzeppelin/contracts/utils/math/SafeMath.sol	7
@openzeppelin/contracts/utils/structs/EnumerableSet.sol	4

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

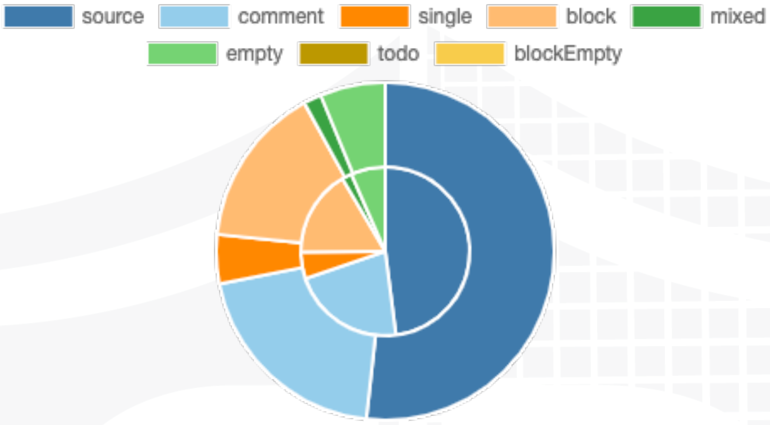


File Name	SHA-1 Hash
contracts/interfaces/IJamonSharePresale.sol	0f0fe57831b17ffe9736313883f0379c1666596c
contracts/interfaces/IConversor.sol	c4d2abb95bd34c3d17e3c403d88d11bf6e756136
contracts/interfaces/IJamonVesting.sol	c9956a482deb7e04240129136a9a2928a33b984f
contracts/interfaces/IJamonShareVesting.sol	6b49e665293c6ab006c2ec4d1fb2c68618e3aa46
contracts/interfaces/IERC20MintBurn.sol	304ea9d0e6ffaebe6fc6cf9a004ad55fd31235d6
contracts/interfaces/IJamonRouter.sol	a38638b24ff3b67fdf8bf37368b796e5900e1dac
contracts/interfaces/IJamonPair.sol	0efc85f7bf717969dd5e8a1e7faf22b5b613a1f8
contracts/tokens/JamonShare.sol	3b7c957fd74c78c7767a9ede34729a3bf09c82e5
contracts/tokens/JamonV2.sol	d1e532fb8bc8fcf9d6c4e97c7abd964161fc6905
contracts/JamonVesting.sol	5c74ea00b3fc96e6ec41d3a55f3bd9fb2c04d897
contracts/JamonShareVault.sol	9f253d6b5ecad903bc90f2ff12f6cacfd91356e5
contracts/Conversor.sol	a5d075b4f0897e07ede25135d2b11f338e359e9e
contracts/JamonSharePresale.sol	a4858bde72be967f138a2dc40fa55f730164d965
contracts/Bonus.sol	d737bc58ee04f210475c4bb1640297ed8bcbbd05
contracts/JamonShareVesting.sol	8fff8ac326aa9d5034054dbbbae5b559c40e4dff
contracts/JamonGovernor.sol	7e36ffc32cce51f19e1b059e3748407d28d32899
contracts/JamonVault.sol	9b6a496d16d29d780fbb1e82215e16db6e54deee

Metrics

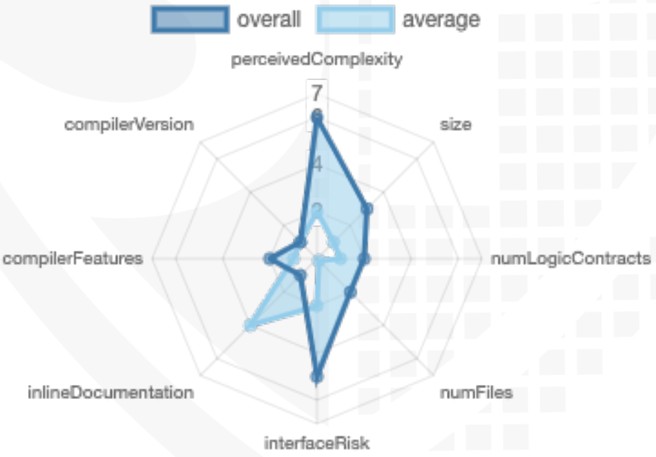
Source Lines

v1.0



Risk Level

v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	10	0	8	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	122	0

Version	External	Internal	Private	Pure	View
1.0	87	142	0	3	53

State Variables

Version	Total	Public
1.0	73	19

Capabilities

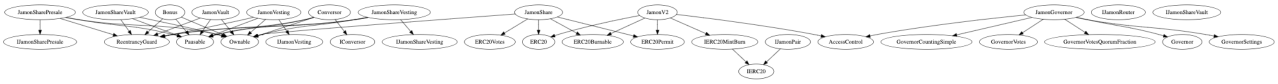
Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	^0.8.11				

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

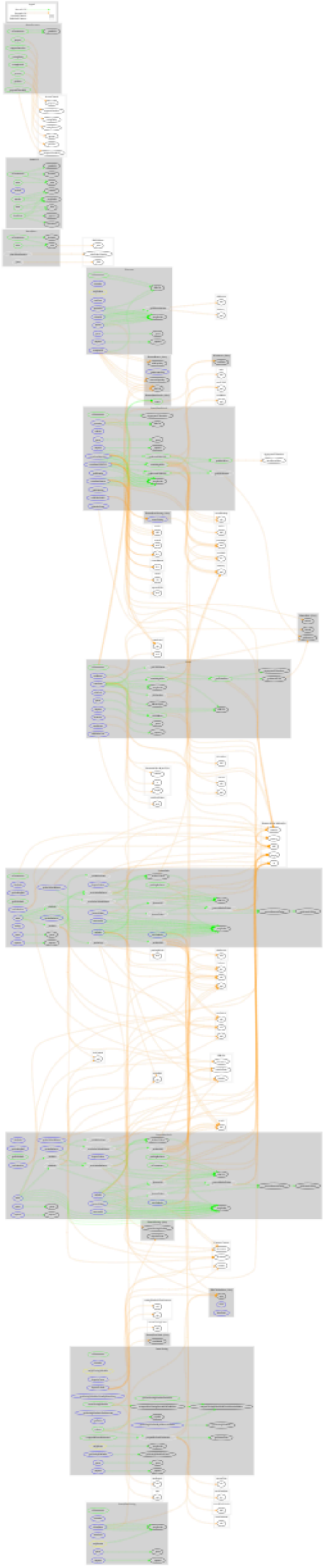
1.0	yes			yes		

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).

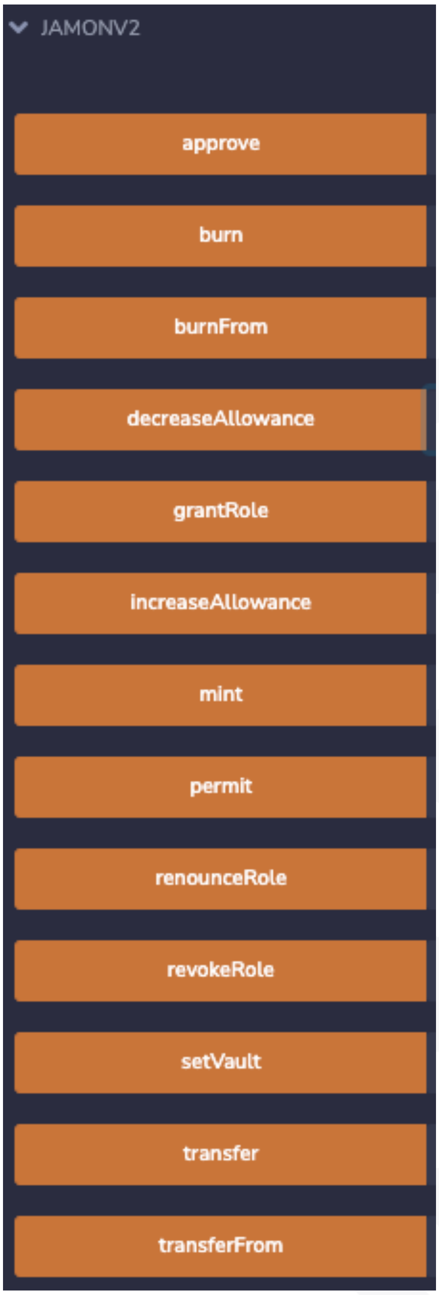
We will verify the following claims:

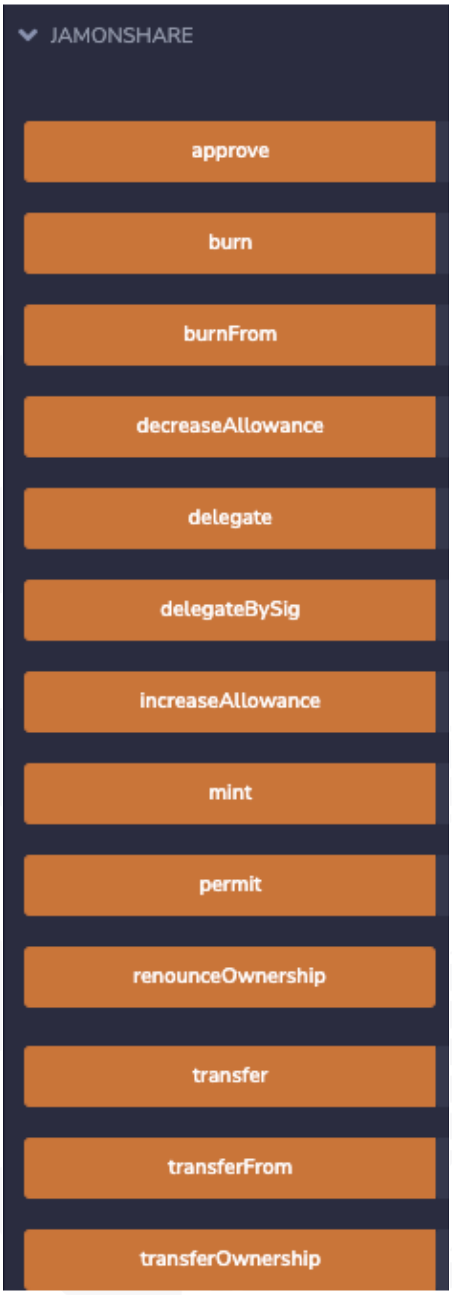
1. Correct implementation of Token standard
2. Cannot mint any new tokens
3. Cannot burn or lock user funds
4. Cannot pause the contract
5. Overall checkup (Smart Contract Security)

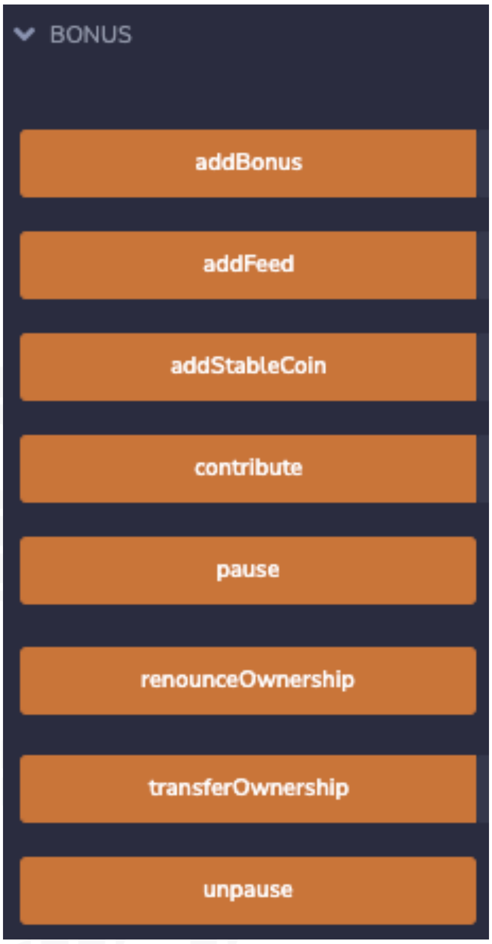
Correct implementation of Token standard

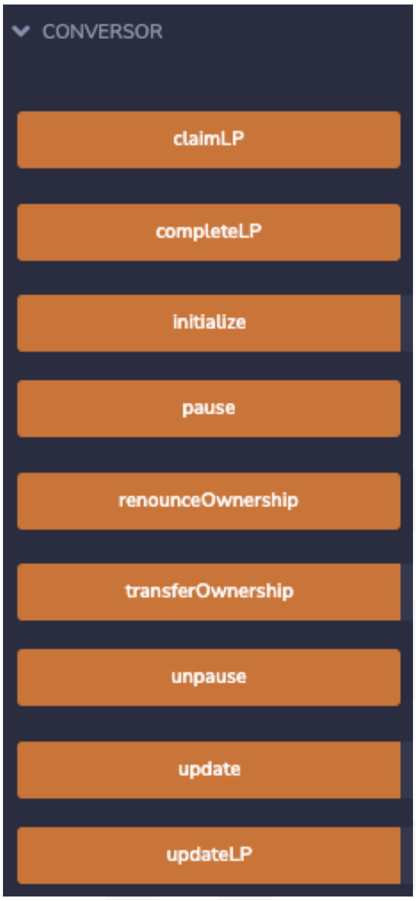
Function	Description	Exist	Tested	Verified
TotalSupply	provides information about the total token supply	✓	✓	✓
BalanceOf	provides account balance of the owner's account	✓	✓	✓
Transfer	executes transfers of a specified number of tokens to a specified address	✓	✓	✓
TransferFrom	executes transfers of a 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	✓	✓	✓

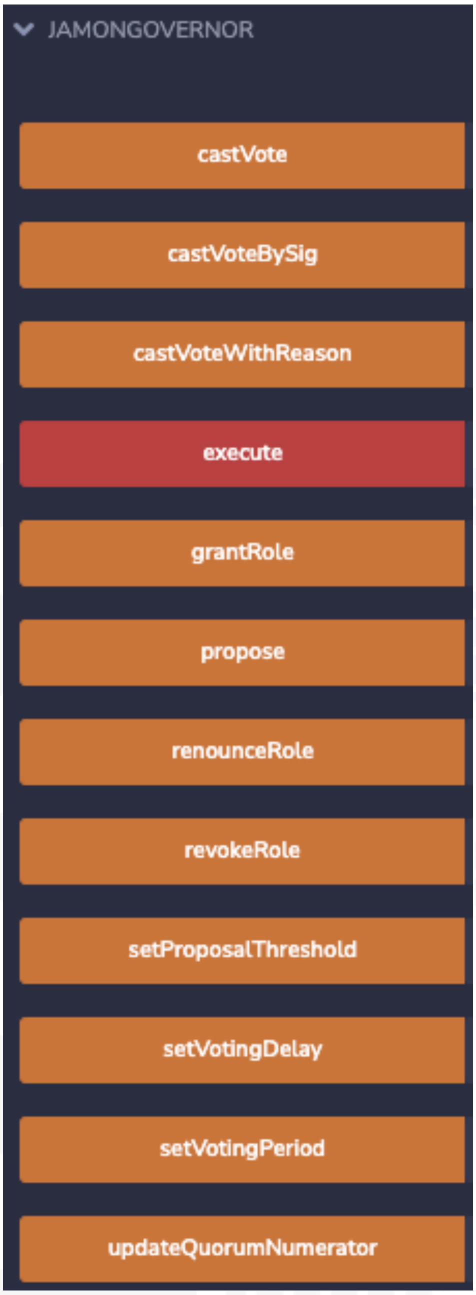
Write functions of contract v1.0

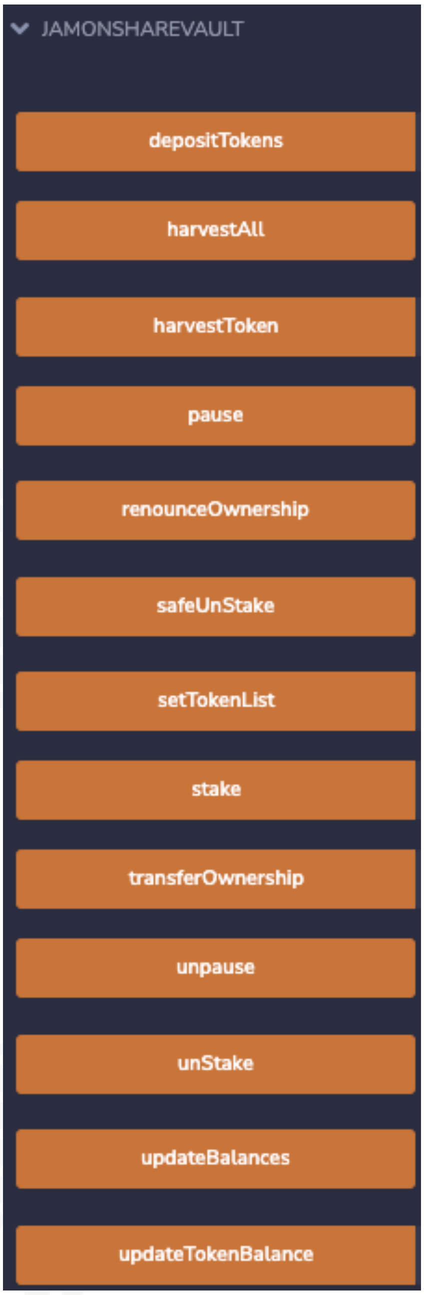


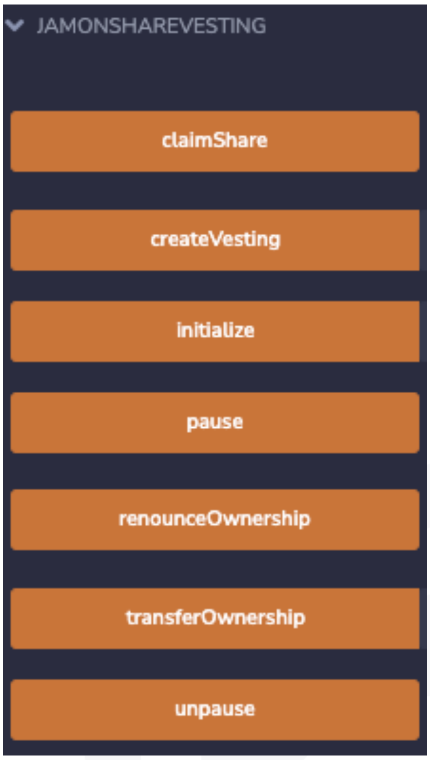


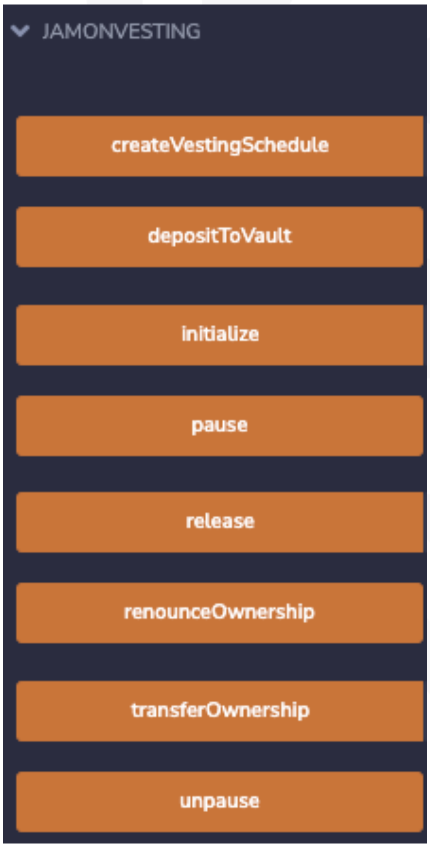


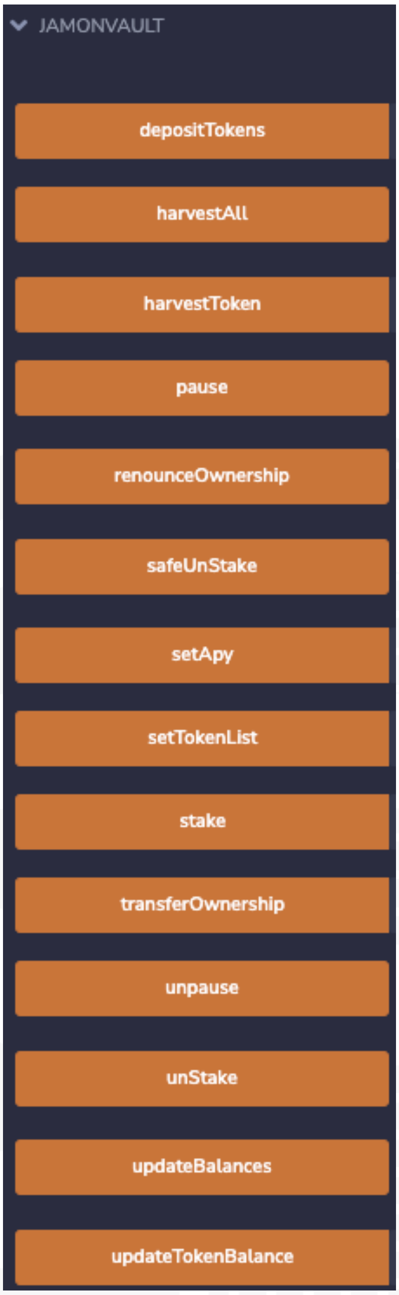

CONVERSOR
claimLP
completeLP
initialize
pause
renounceOwnership
transferOwnership
unpause
update
updateLP

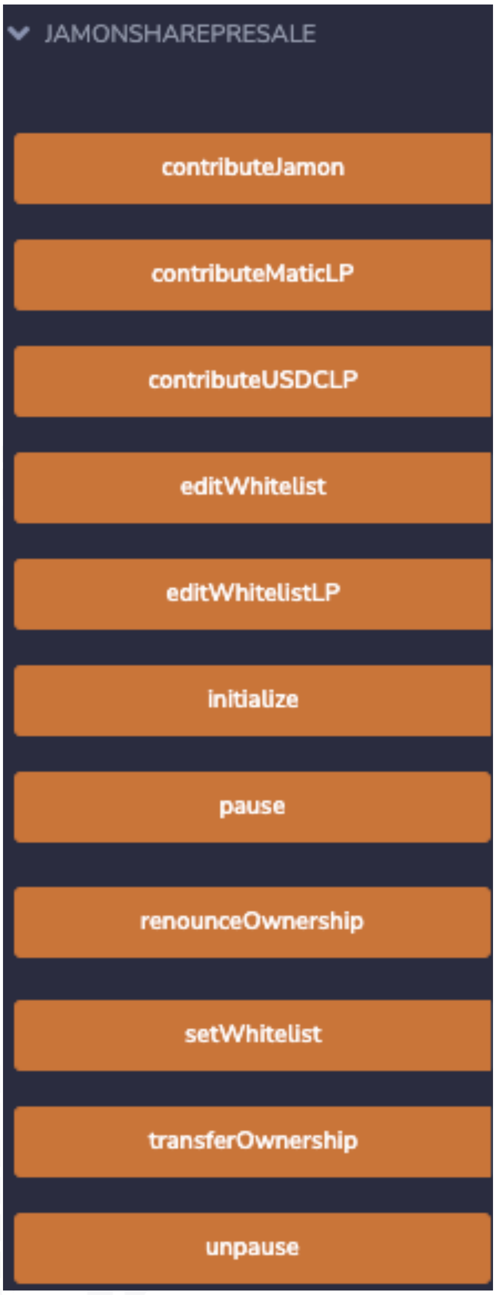

JAMONGOVERNOR
castVote
castVoteBySig
castVoteWithReason
execute
grantRole
propose
renounceRole
revokeRole
setProposalThreshold
setVotingDelay
setVotingPeriod
updateQuorumNumerator


JAMONSHAREVAULT
depositTokens
harvestAll
harvestToken
pause
renounceOwnership
safeUnStake
setTokenList
stake
transferOwnership
unpause
unStake
updateBalances
updateTokenBalance


JAMONSHAREVESTING
claimShare
createVesting
initialize
pause
renounceOwnership
transferOwnership
unpause


JAMONVESTING
createVestingSchedule
depositToVault
initialize
pause
release
renounceOwnership
transferOwnership
unpause


JAMONVAULT
depositTokens
harvestAll
harvestToken
pause
renounceOwnership
safeUnStake
setApy
setTokenList
stake
transferOwnership
unpause
unStake
updateBalances
updateTokenBalance


JAMONSHAREPRESALE
contributeJamon
contributeMaticLP
contributeUSDCLP
editWhitelist
editWhitelistLP
initialize
pause
renounceOwnership
setWhitelist
transferOwnership
unpause

Cannot mint any new tokens

Name	Exist	Tested	Status
Cannot mint	✓	✓	✗

Comments:

v1.0

- Conversor
    - update function will mint new tokens (convert the old token for the new one
    - onlyOwner can call completeLP function to mint new token (change old liquidity to new liquidity)
- JamonShareVesting
    - claimShare will mint new tokens
    - createVesting can only be called from presale and mint new tokens
- JamonVault
    - updateBalances will mint new tokens
- JamonVesting
    - Release will mint new tokens (only if vesting schedules)
    - depositToVault will mint new tokens to shareVault address

- Every contract which is using IERC20MintBurn interface can mint/burn

Cannot burn or lock user funds

Name	Exist	Tested	Status
Cannot lock	✓	✓	✗
Cannot burn	✓	✓	✗

Comments:

v1.0

- JamonSharePresale
  - contributeJamon function will burn from caller
  - onlyOwner can lock contributeMaticLP function if round equal to 1 and user is not in WhitelistLP
  - Deployer can lock contributeJamon function by setting a Max4Wallet amount to an specific address

- Every contract which is using IERC20MintBurn interface can mint/burn

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	✓	✓	✗

Comments:

v1.0

- Deployer can lock following functions with pausing the contract
  - Bonus
    - contribute
  - Conversor
    - updateLP
    - Update
    - claimLP
  - JamonSharePresale
    - contributeMaticLP
    - contributeUSDCLP
    - contributeJamon
  - JamonShareVault
    - Stake
    - harvestToken
    - harvestAll
    - unStake
    - updateBalance
  - JamonShareVesting
    - createVesting
  - JamonVault
    - Stake
    - harvestToken
    - harvestAll
    - unStake
    - updateBalance
  - JamonVesting
    - createVestingSchedule
    - Release
    - depositToVault

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

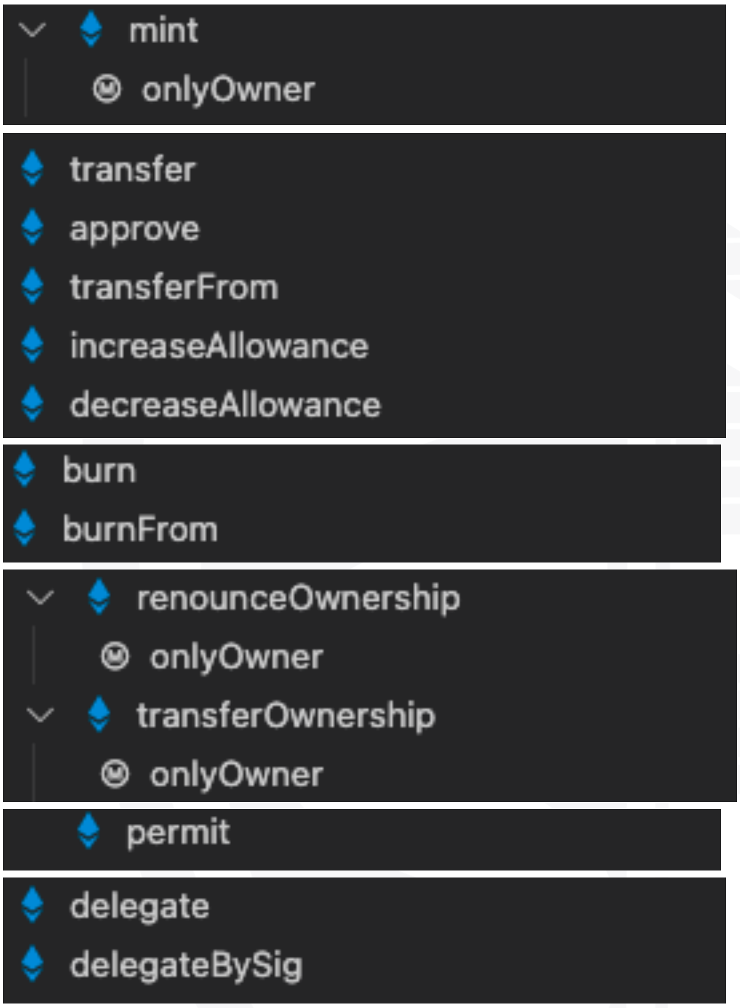
Legend

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

Modifiers and public functions

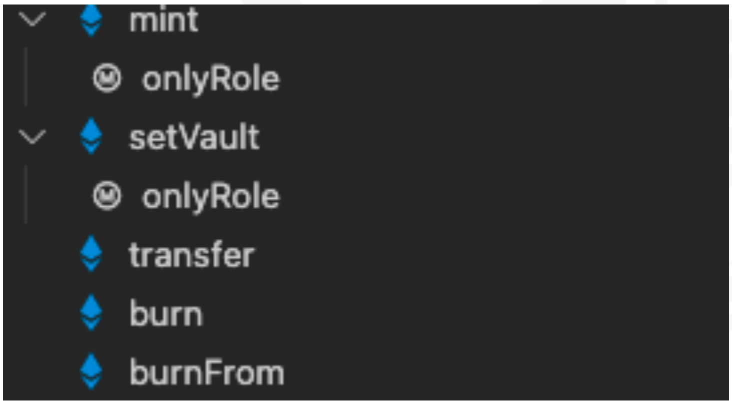
v1.0

JamonShare



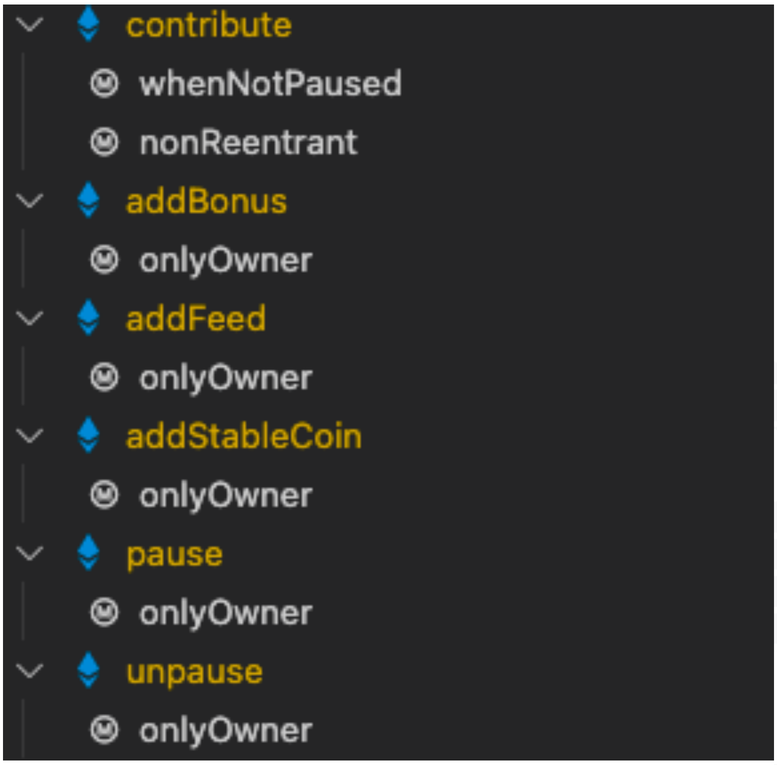
```
mint
    onlyOwner
transfer
approve
transferFrom
increaseAllowance
decreaseAllowance
burn
burnFrom
renounceOwnership
    onlyOwner
transferOwnership
    onlyOwner
permit
delegate
delegateBySig
```

JamonV2



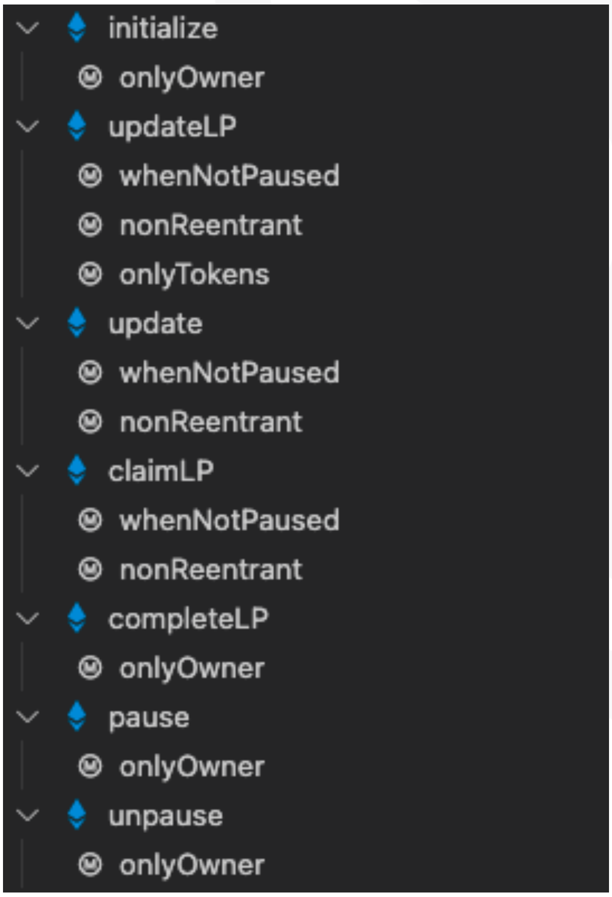
```
mint
    onlyRole
setVault
    onlyRole
transfer
burn
burnFrom
```

Bonus



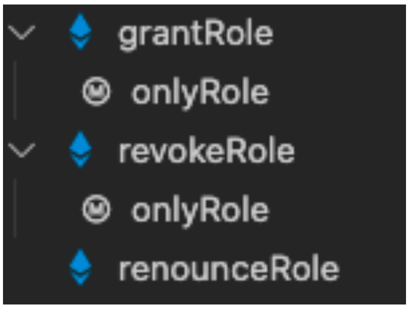
```
contribute
    whenNotPaused
    nonReentrant
addBonus
    onlyOwner
addFeed
    onlyOwner
addStableCoin
    onlyOwner
pause
    onlyOwner
unpause
    onlyOwner
```

Conversor

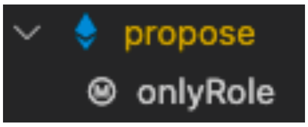


```
initialize
    onlyOwner
updateLP
    whenNotPaused
    nonReentrant
    onlyTokens
update
    whenNotPaused
    nonReentrant
claimLP
    whenNotPaused
    nonReentrant
completeLP
    onlyOwner
pause
    onlyOwner
unpause
    onlyOwner
```

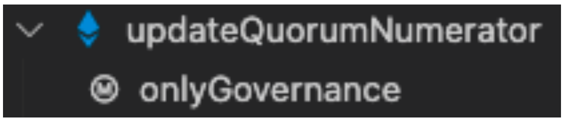
JamonGovernor



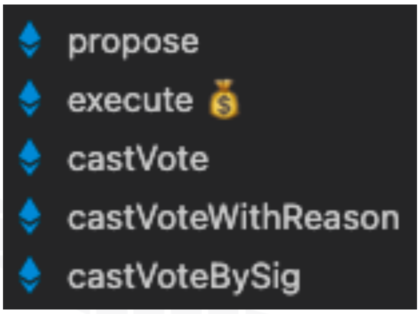
```
grantRole
    onlyRole
revokeRole
    onlyRole
renounceRole
```



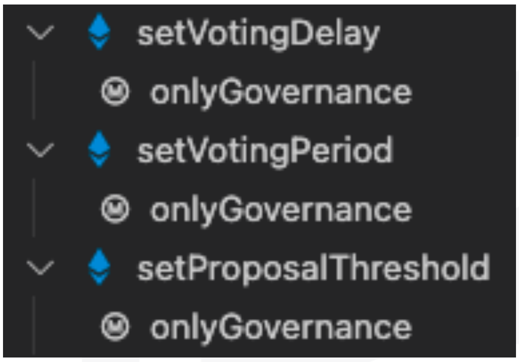
```
propose
    onlyRole
```



```
updateQuorumNumerator
    onlyGovernance
```

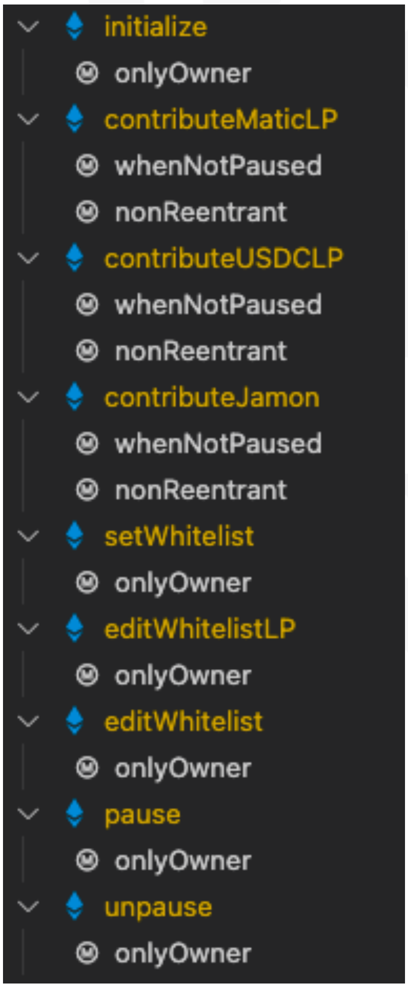


```
propose
execute 💰
castVote
castVoteWithReason
castVoteBySig
```



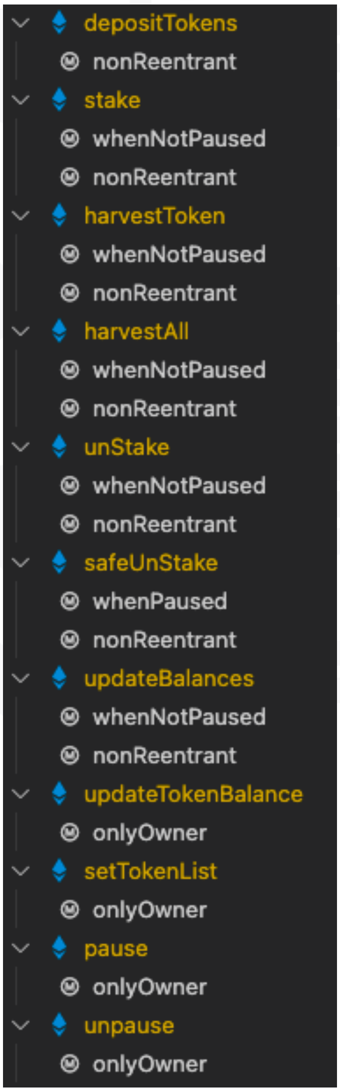
```
setVotingDelay
    onlyGovernance
setVotingPeriod
    onlyGovernance
setProposalThreshold
    onlyGovernance
```

JamonSharePresale



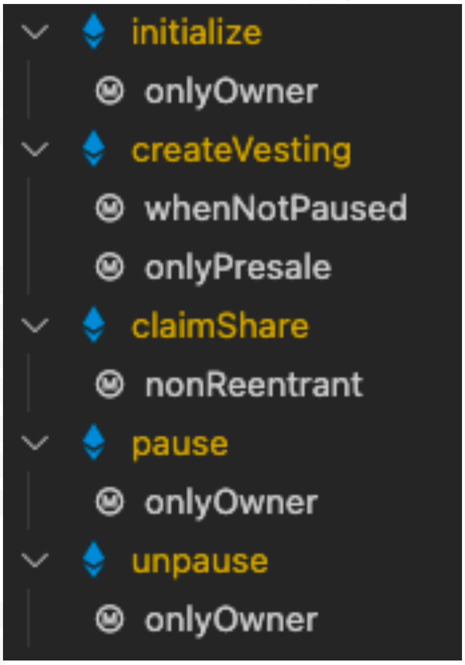
```
initialize
    onlyOwner
contributeMaticLP
    whenNotPaused
    nonReentrant
contributeUSDCLP
    whenNotPaused
    nonReentrant
contributeJamon
    whenNotPaused
    nonReentrant
setWhitelist
    onlyOwner
editWhitelistLP
    onlyOwner
editWhitelist
    onlyOwner
pause
    onlyOwner
unpause
    onlyOwner
```

JamonShareVault



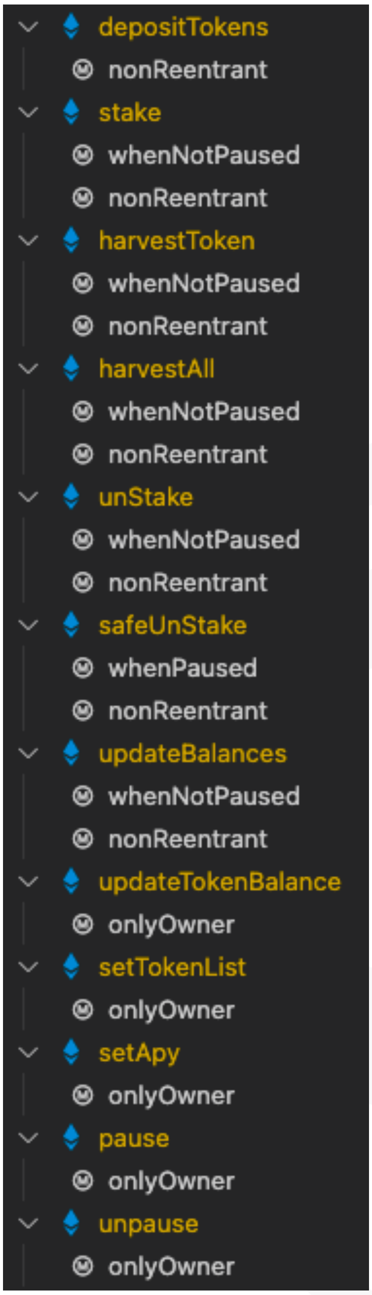
```
depositTokens
    nonReentrant
stake
    whenNotPaused
    nonReentrant
harvestToken
    whenNotPaused
    nonReentrant
harvestAll
    whenNotPaused
    nonReentrant
unStake
    whenNotPaused
    nonReentrant
safeUnStake
    whenPaused
    nonReentrant
updateBalances
    whenNotPaused
    nonReentrant
updateTokenBalance
    onlyOwner
setTokenList
    onlyOwner
pause
    onlyOwner
unpause
    onlyOwner
```

JamonShareVesting



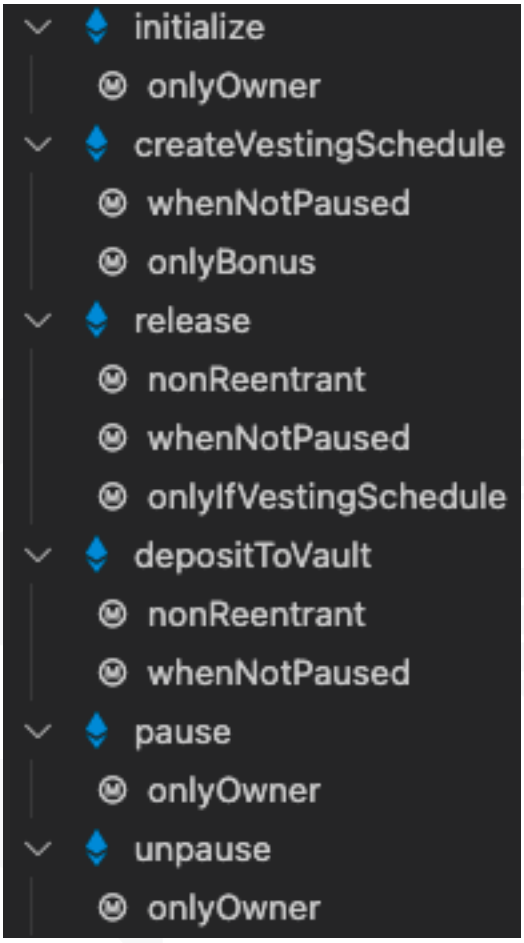
```
initialize
    onlyOwner
createVesting
    whenNotPaused
    onlyPresale
claimShare
    nonReentrant
pause
    onlyOwner
unpause
    onlyOwner
```

JamonVault



```
depositTokens
    nonReentrant
stake
    whenNotPaused
    nonReentrant
harvestToken
    whenNotPaused
    nonReentrant
harvestAll
    whenNotPaused
    nonReentrant
unStake
    whenNotPaused
    nonReentrant
safeUnStake
    whenPaused
    nonReentrant
updateBalances
    whenNotPaused
    nonReentrant
updateTokenBalance
    onlyOwner
setTokenList
    onlyOwner
setApy
    onlyOwner
pause
    onlyOwner
unpause
    onlyOwner
```

JamonVesting



```
initialize
    onlyOwner
createVestingSchedule
    whenNotPaused
    onlyBonus
release
    nonReentrant
    whenNotPaused
    onlyIfVestingSchedule
depositToVault
    nonReentrant
    whenNotPaused
pause
    onlyOwner
unpause
    onlyOwner
```

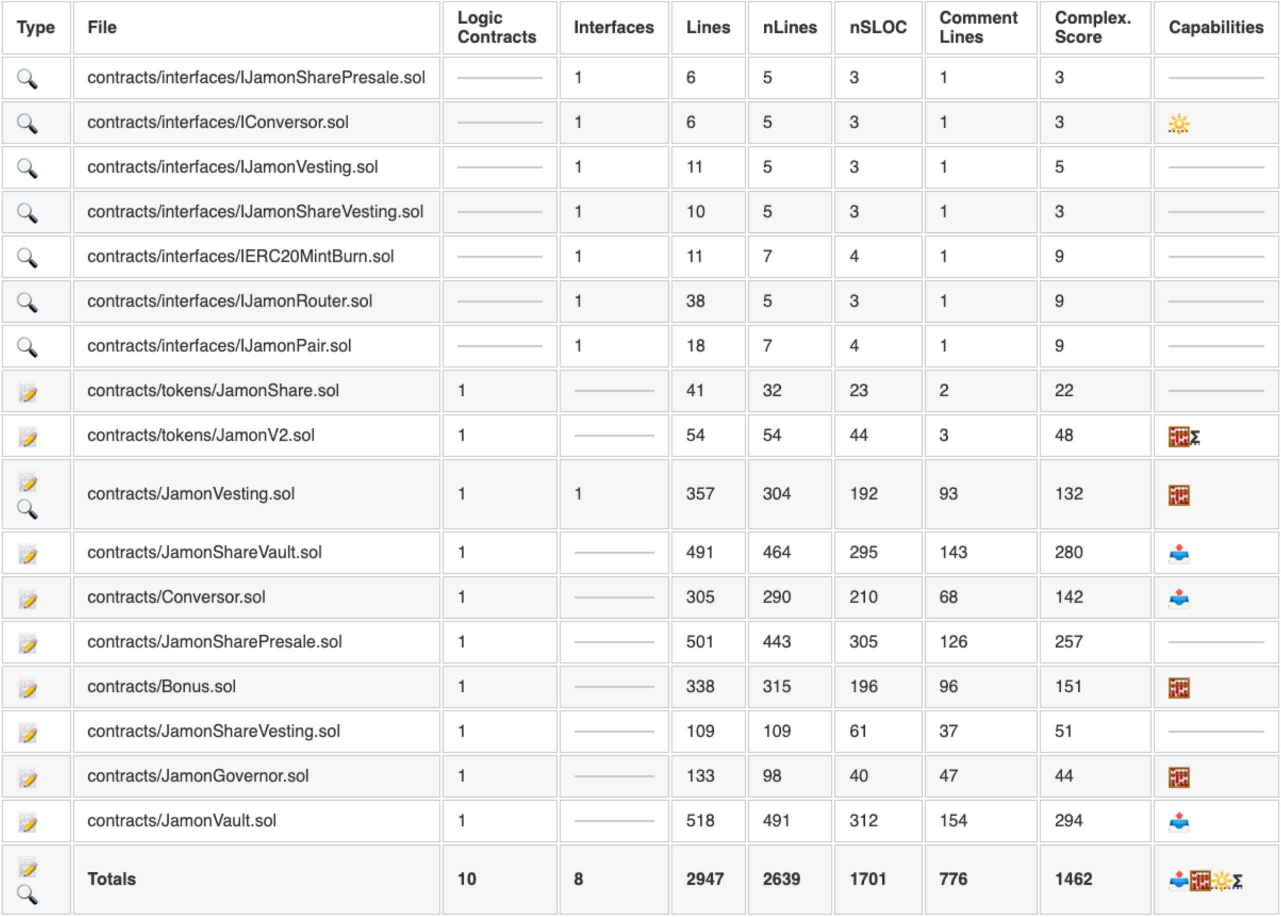
Comments

- Deployer can set following state variables without any limitations
    - JamonShare
        - _totalSupply
    - JamonVesting
        - VestingSchedule amountTotal
    - Bonus
        - Proposal
            - startBlock_
            - endBlock
            - hardcap
        - FEEDS[token_].decimals
    - JamonShareVesting
        - Mint/SHATE_VESTING
- Deployer can enable/disable following state variables
    - JamonVesting
        - _paused
    - JamonSharePresale
        - listActive
        - WhitelistLP
        - Max4Wallet

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/interfaces/IJamonSharePresale.sol	—	1	6	5	3	1	3	—
🔍	contracts/interfaces/IConversor.sol	—	1	6	5	3	1	3	🔆
🔍	contracts/interfaces/IJamonVesting.sol	—	1	11	5	3	1	5	—
🔍	contracts/interfaces/IJamonShareVesting.sol	—	1	10	5	3	1	3	—
🔍	contracts/interfaces/IERC20MintBurn.sol	—	1	11	7	4	1	9	—
🔍	contracts/interfaces/IJamonRouter.sol	—	1	38	5	3	1	9	—
🔍	contracts/interfaces/IJamonPair.sol	—	1	18	7	4	1	9	—
📝	contracts/tokens/JamonShare.sol	1	—	41	32	23	2	22	—
📝	contracts/tokens/JamonV2.sol	1	—	54	54	44	3	48	🧮Σ
📝 🔍	contracts/JamonVesting.sol	1	1	357	304	192	93	132	🧮
📝	contracts/JamonShareVault.sol	1	—	491	464	295	143	280	📥
📝	contracts/Conversor.sol	1	—	305	290	210	68	142	📥
📝	contracts/JamonSharePresale.sol	1	—	501	443	305	126	257	—
📝	contracts/Bonus.sol	1	—	338	315	196	96	151	🧮
📝	contracts/JamonShareVesting.sol	1	—	109	109	61	37	51	—
📝	contracts/JamonGovernor.sol	1	—	133	98	40	47	44	🧮
📝	contracts/JamonVault.sol	1	—	518	491	312	154	294	📥
📝 🔍	**Totals**	**10**	**8**	**2947**	**2639**	**1701**	**776**	**1462**	📥🧮🔆Σ

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

Issue	File	Type	Line	Description
#1	JamonSharePresale	Uninitialized state variables	43	Initialize all the variables. If a variable is meant to be initialized to zero, explicitly set it to zero to improve code readability. Every function which is calling Conversor.<functionName>() will fail. Function which is calling Conversor: - Initialize function line: 102 - status function line 187
#2	JamonSharePresale	Uninitialized state variables	50	Initialize all the variables. If a variable is meant to be initialized to zero, explicitly set it to zero to improve code readability. Every function which is using listLimit boolean will fail. Function which is calling listLimit: - contributeMaticLP line: 317 - Contribute USDCLP line: 369

Medium issues

No medium issues

Low issues

Issue	File	Type	Line	Description
#1	JamonShare	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.11””.
#2	JamonV2	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.11””.
#3	Bonus	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.11””.
#4	Conversor	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.11””.
#5	Jamon Governor	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.11””.
#6	JamonSharePresale	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.11””.
#7	JamonShareVault	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.11””.
#8	JamonShareVesting	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.11””.
#9	JamonVault	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.11””.
#10	JamonVesting	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.11””.
#11	Bonus	Missing Zero Address Validation (missing-zero-check)	84, 85	Check that the address is not zero
#12	JamonSharePresale	Missing Zero Address Validation (missing-zero-check)	108	Check that the address is not zero
#13	JamonShareVault	Missing Zero Address Validation (missing-zero-check)	60	Check that the address is not zero

#14	JamonShareVesting	Missing Zero Address Validation (missing-zero-check)	51	Check that the address is not zero
#15	JamonV2	Missing Zero Address Validation (missing-zero-check)	26	Check that the address is not zero
#16	JamonShareVesting	Missing Events Access Control	53	Emit an event for critical parameter changes
#17	JamonVesting	Missing Events Access Control	76	Emit an event for critical parameter changes
#18	JamonVault	Missing Events Arithmetic	505	Emit an event for critical parameter changes

Informational issues

Issue	File	Type	Line	Description
#1	JamonSharePresale	State variables that could be declared constant (constable-states)	50	Add the `constant` attributes to state variables that never change
#2	Bonus	Unused return values	317, 326, 101	Ensure that all the return values of the function calls are used and handle both success and failure cases if needed by the business logic
#3	Conversor	Unused return values	254, 279, 277, 252	Ensure that all the return values of the function calls are used and handle both success and failure cases if needed by the business logic
#4	JamonSharePresale	Unused return values	462, 466	Ensure that all the return values of the function calls are used and handle both success and failure cases if needed by the business logic
#5	JamonShareVault	Unused return values	63, 64, 476, 478	Ensure that all the return values of the function calls are used and handle both success and failure cases if needed by the business logic

#6	JamonVault	Unused return values	65, 492, 494	Ensure that all the return values of the function calls are used and handle both success and failure cases if needed by the business logic
#7	JamonShareVesting	Unused state variables	30	Remove unused state variables
#8	JamonShare	NatSpec documentation missing	-	If you start to comment your code, also comment all other functions, variables etc.
#9	JamonV2	NatSpec documentation missing	-	If you start to comment your code, also comment all other functions, variables etc.
#10	Bonus	Misspelling	Line next to change	Change following word: - acepted to accepted line: 270 If word is function/variable etc. make sure to change it everywhere else
#11	Jamon Governor	Misspelling	Line next to change	Change following word: - increassed to increased line: 33 If word is function/variable etc. make sure to change it everywhere else
#12	JamonSharePresale	Misspelling	Line next to change	Change following word: - linmit to limit line: 50 If word is function/variable etc. make sure to change it everywhere else
#13	JamonShareVault	Misspelling	Line next to change	Change following word: - reawrds to rewards lines: 353 If word is function/variable etc. make sure to change it everywhere else

#14	Bonus	Require error message is missing	285, 286, 287, 288, 290, 313, 314, 325	Provide an error message for the require statement
#15	JamonSharePresale	Require error message is missing	324, 376	Provide an error message for the require statement
#16	Conversor	Require error message is missing	120, 235	Provide an error message for the require statement
#17	JamonShareVault	Require error message is missing	81, 380, 387, 389, 422,440, 470	Provide an error message for the require statement
#18	JamonShareVesting	Require error message is missing	61	Provide an error message for the require statement
#19	JamonVault	Require error message is missing	85, 399, 406, 408, 441, 459, 490, 504	Provide an error message for the require statement
#20	JamonVesting	Require error message is missing	65, 85, 93, 338, 339	Provide an error message for the require statement
#21	JamonV2	Require error message is missing	27	Provide an error message for the require statement

Audit Comments

27. January 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	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