SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

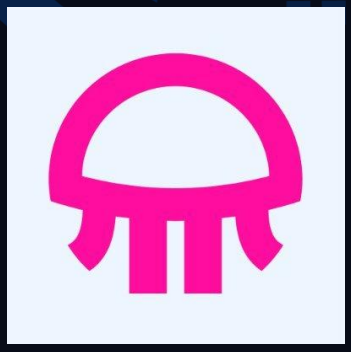
jAssets

by BLKSWN PTE. LTD.

AUDIT

SECURITY ASSESSMENT

10. October, 2024



Introduction

SolidProof.io is a brand of the officially registered company MAKE Network GmbH, based in Germany. We're mainly focused on Blockchain Security such as Smart Contract Audits and KYC verification for project teams.
Solidproof.io assess potential security issues in the smart contracts implementations, review for potential inconsistencies between the code base and the whitepaper/documentation, and provide suggestions for improvement.

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 the security or functionality of the technology we agree to analyze.

Project Overview

Summary

Project Name	jAssets
Website	https://jellyverse.org/
About the project	jAssets is a collateralized debt platform. Users can lock up collateral (specific ERC20 tokens, selected by the governance) and issue jAssets (jUSD, jAAPL, jTSLA, etc.) to their own address and subsequently transfer those tokens to any other address. The individual collateralized debt positions are called Troves.
Chain	Sei
Language	Solidity
Codebase	Provided as file.
Commit	N/A
Unit Tests	Not Provided

Social Medias

Telegram	N/A
Twitter	https://x.com/BLKSWN_STUDIO
Facebook	https://www.facebook.com/jlyvrs
Instagram	N/A
GitHub	N/A
Reddit	N/A
Medium	N/A
Discord	https://www.reddit.com/r/jellyverse/
YouTube	N/A
TikTok	N/A
LinkedIn	https://www.linkedin.com/company/jlyvrs

Audit Summary

Version	Delivery Date	Change Log
v1.0	09. October 2024	· Layout Project · Automated/Manual-Security Testing · Summary

Note – The following audit report presents a comprehensive security analysis of the smart contract utilized in the project that includes outside manipulation of the contract's functions in a malicious way. This analysis did not include functional testing (or unit testing) of the contract/s logic. We cannot guarantee 100% logical correctness of the contract as we did not functionally test it. This includes internal calculations in the formulae used in the contract.

File Overview

The Team provided us with the files that should be tested in the security assessment. This audit covered the following files listed below with an SHA-1 Hash.

File Name	SHA-1 Hash
contracts/PriceFeed.sol	7ee482094a32f301952cc7483e432ea0d121b0d0
contracts/LiquidationOperations.sol	c01d7824c556202a7a907a23cda22a03e8641b93
contracts/TroveManager.sol	c2e714abeb4eb260b391cc0dec34e8e92b42c04d
contracts/BorrowerOperations.sol	3403f95e05c2aba3db792652304aae7849ff16c4
contracts/StabilityPoolManager.sol	57f057c683dead92da2471911406e3aab5604154
contracts/StabilityPool.sol	cc03205a14254fdb94d5f32fc310e9bc8223ad53
contracts/StoragePool.sol	7368f6c1e69e6f5bb2ff4ff42bc9399f62105934
contracts/SwapPair.sol	9c64077bc0adbd2bbd61334da1c9df91d9a5dd8d
contracts/AlternativePriceFeed.sol	3d930c47ab11e10d3df2207c78eabe9405e4e6c7
contracts/CollSurplusPool.sol	2680a6c2242633dc94775bd97dbbb45bd70d632f
contracts/Dependencies/LiquityMath.sol	3dfc1d3c78e509fcf87c19385ac0d3a60ba656ea
contracts/Dependencies/UQ112x112.sol	c68778896cff3c54fab3ec03bd908e9bb4fb56da
contracts/Dependencies/LiquityBase.sol	3c1232390d251ca70ec3eb08e5d6152571accdc9
contracts/Dependencies/TrustlessPermit.sol	518c5c9fa3c237a101bd88e2111fbddfd402876c
contracts/Dependencies/IERC2612.sol	7b03731c955d0436344bc90f22c9b4b8ef01404c
contracts/Dependencies/CheckContract.sol	33e55e088c2601783ebe39cb32cf7a36755b39f5
contracts/Dependencies/IBalancerV2Pool.sol	2b986f76d8ada2bee01093eb520bfb0b3e1e5c5c
contracts/Dependencies/IBalancerV2Vault.sol	9ebcb1960d25cd6e9500ce1345fd7161da8cf70f
contracts/TokenManager.sol	622bfe09322f676e3dbd9105f83a2b39741924a1
contracts/SwapERC20.sol	7156154b39566b985f46f2df926fe4def6f31b53
contracts/SwapOperations.sol	d7d895e126e47079615220f8e0ef179189485759
contracts/StakingOperations.sol	ca34835a855ea5359b9fbab9b19ac74b868067e7

File Name	SHA-1 Hash
contracts/RedemptionOperations.sol	8a34fd1be78e8a0ee38c1404a672c7a699b56dae
contracts/SortedTroves.sol	a743ede79afe9d2f46c1a5537b75359f861e7156
contracts/DebtToken.sol	736fe45818e37f7c2a823d196369f3446592202f
contracts/ReservePool.sol	e464b3d7e0443f7ed8408a02736d30e8204f3712
contracts/HintHelpers.sol	ed4367457d7c145e0674a2f0c55b7df2702d4928

Please note: Files with a different hash value than in this table have been modified after the security check, either intentionally or unintentionally. A different hash value may (but need not) be an indication of a changed state or potential vulnerability that was not the subject of this scan.

Imported packages.

Used code from other Frameworks/Smart Contracts.

Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable.sol	15
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	10
@openzeppelin/contracts/token/ERC20/extensions/ERC20Permit.sol	1
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	8
@openzeppelin/contracts/token/ERC20/extensions/IERC20Permit.sol	1
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	6
@openzeppelin/contracts/utils/Address.sol	1
@openzeppelin/contracts/utils/cryptography/ECDSA.sol	1
@openzeppelin/contracts/utils/cryptography/MessageHashUtils.sol	1
@openzeppelin/contracts/utils/math/Math.sol	1
@pythnetwork/pyth-sdk-solidity/IPyth.sol	1
@pythnetwork/pyth-sdk-solidity/MockPyth.sol	1
@pythnetwork/pyth-sdk-solidity/PythStructs.sol	1

Note for Investors: We only audited contracts mentioned in the scope above. All contracts related to the project apart from that are not a part of the audit, and we cannot comment on its security and are not responsible for it in any way.

External/Public functions

External/public functions are functions that can be called from outside of a contract, i.e., they can be accessed by other contracts or external accounts on the blockchain. These functions are specified using the function declaration's external or public visibility modifier.

State variables

State variables are variables that are stored on the blockchain as part of the contract'sstate. They are declared at the contract level and can be accessed and modified by any function within the contract. State variables can be needed within visibility modifier, such as public, private or internal, which determines the access level of the variable.

Components

📝 Contracts	📚 Libraries	🔍 Interfaces	🎨 Abstract
29	3	26	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.

🌐 Public	💰 Payable
521	45

External	Internal	Private	Pure	View
485	389	7	53	320

StateVariables

Total	🌐 Public
214	181

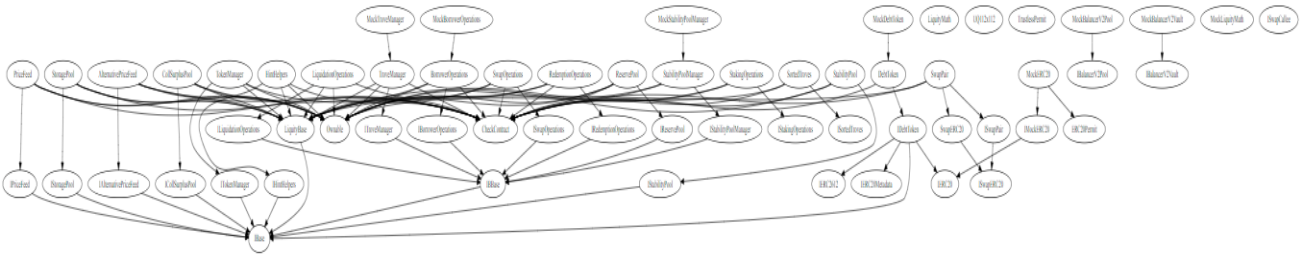
Capabilities

Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
`^0.8.9`	`----------`	`Yes`	`yes (3 asm blocks)`	`----------`

Transfers ETH	Low-Level Calls	Delegate Call	Uses Hash Functions	ECRecover	New/Create/Create2
			`Yes`		`yes → NewContract:StabilityPool`

Inheritance Graph

An inheritance graph is a graphical representation of the inheritance hierarchy among contracts. In object-oriented programming, inheritance is a mechanism that allows one class (or contract, in the case of Solidity) to inherit properties and methodsfrom another class. It shows the relationships between different contracts and how they are related to each other through inheritance.



Audit Information

Vulnerability & Risk Level

Risk represents the probability that a certain source threat will exploit the vulnerability and the impact of that event on the organization or system.The 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 executingthe 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 check the repository for security-related issues, code quality, and compliance with specifications and best practices. To this end, our team of experienced pen-testers andsmart contract developers reviewed the code line by line and documented any issues discovered.

We check every file manually. We use automated tools only so that they help us achieve faster and better results.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
   a. Reviewing the specifications, sources, and instructions provided to SolidProof to ensure we understand the size, scope, and functionality of the smart contract.
   b. Manual review of the code, i.e., reading the source code line by line to identify potential vulnerabilities.
   c. Comparison to the specification, i.e., verifying that the code does what is described in the specifications, sources, and instructions provided to SolidProof.

2. Testing and automated analysis that includes the following:
   a. Test coverage analysis determines whether test cases cover code and how much code is executed when those test cases are executed.
   b. Symbolic execution, which is analysing a program to determine what inputs cause each part of a program to execute.

3. Review best practices, i.e., review smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on best practices, recommendations, and research from industry and academia.

4. Concrete, itemized and actionable recommendations to help you secure your smart contracts.

Overall Security

Upgradeability

Contract is not upgradable	✅ **Deployer cannot update the contract with new functionalities.**
Description	The contract is not an upgradeable contract. The Deployer is not able to change or add any functionalities to the contract after deploying.
Comment	N/A

Ownership

Contract ownership is not renounced.	❌ **The ownership is not renounced.**
Description	The owner has not renounced the ownership that means that the owner retains control over the contract's operations, including the ability to execute functions that may impact the contract's users or stakeholders. This can lead to several potential issues, including: • Centralizations • The owner has significant control over contract's operations.
Comment	N/A

Note – *The contract cannot be considered as renounced till it is not deployed or having some functionality that can change the state of the contract.*

Ownership Privileges

These functions can be dangerous. Please note that abuse can lead to financial loss. We have a guide where you can learn more about these Functions.

Minting tokens

Minting tokens refer to the process of creating new tokens in a cryptocurrency or blockchain network. This process is typically performed by the project's owner or designated authority, who has the ability to add new tokens to the network's total supply.

Contract owner can mint new tokens.	❌ **The owner can mint new tokens.**
Description	The owner is able to mint new tokens once the contract is deployed.
Comment	The contract contains the functionality in which the trove manager, borrower operation of the contract can mint an unlimited amount of tokens to any wallet which is not recommended as this can manipulate the total supply of the token and price of the token will also be fluctuated because of this functionality. There must be a fixed max total supply so that the tokens total supply will not be changed.

File/Line(s): L190-194

Codebase: DebtToken.sol

```
function mint(address _account, uint256 _amount) external override {
    _requireCallerIsBorrowerOperationsOrTroveManager();
    _requireMintingEnabled();
    _mint(_account, _amount);
}
```

Burning tokens

Burning tokens is the process of permanently destroying a certain number of tokens, reducing the total supply of a cryptocurrency or token. This is usually done to increase the value of the remaining tokens, as the reduced supply can create scarcity and potentially drive up demand.

Contract owner cannot burn tokens	✅ **The owner cannot burn tokens.**
Description	The owner is able burn tokens without any allowances.
Comment	N/A

Blacklist addresses

Blacklisting addresses in smart contracts is the process of adding a certain address to a blacklist, effectively preventing them from accessing or participating in certain functionalities or transactions within the contract. This can be useful in preventing fraudulent or malicious activities, such as hacking attempts or money laundering.

Contract owner cannot blacklist addresses.	✅ **The owner cannot blacklist wallets.**
Description	The owner cannot blacklist wallets from transferring tokens.
Comment	N/A

Fees and Tax

In some smart contracts, the owner or creator of the contract can set fees for certain actions or operations within the contract. These fees can be used to cover the cost of running the contract, such as paying for gas fees or compensating the contract's owner for their time and effort indeveloping and maintaining the contract.

Contract owner can levy high taxes	❌ **The owner can set fees more than 25%.**
Description	The owner can set fees of more than 25%.
Comment	The contract contains the functionality in which the owner can update the fees by more than 25% in the contract which is not recommended as the fees in the contract should not be more than 25% in the contract there must be a check so that the value cannot be more than 25%.

File/Line(s): L113-116

Codebase: SwapOperations.sol

```
function setGovSwapFee(uint _govSwapFee) external onlyOwner {
  if (_govSwapFee > DECIMAL_PRECISION) revert FeeExceedMaxPercentage();
  govSwapFee = _govSwapFee;
}
```

File/Line(s): L144-148

Codebase: TroveManager.sol

```
function setBorrowingFeeFloor(uint _borrowingFeeFloor) external onlyOwner {
  if (_borrowingFeeFloor > DECIMAL_PRECISION) revert InvalidParameter();
  borrowingFeeFloor = _borrowingFeeFloor;
  emit SetBorrowingFeeFloor(borrowingFeeFloor);
}
```

Lock User Funds

In a smart contract, locking refers to the process of restricting access to certain tokens or assets for a specified period of time. When token or assets are locked in a smart contract, they cannot be transferred or used until the lock-up period has expired or certain conditions have been met.

Contract owner cannot lock function.	✅ **The owner cannot lock function.**
Description	The owner cannot lock the contract.
Comment	N/A

Centralization Privileges

Centralization can arise when one or more parties have privileged access or control over the contract's functionality, data, or decision-making. This can occur, for example,if the contract is controlled by a single entity or if certain participants have special permissions or abilities that others do not.

In the project, there are authorities that have access to the following functions:

File	Privileges
AlternativePriceFeed	➢ The owner can update the price feed address only once. ➢ The owner can update the fallback price in the contract. ➢ The owner can update the fallback trusted timespan for the token in the contract. ➢ The owner can update the balancer price pool for the token.
BorrowerOperations	➢ The owner can update the trove manager, storage pool, stability pool, price feed, token manager, swap operations, troves, and Coll surplus contract address.
CollSurplusPool	➢ The owner can update the liquidation and borrower operation contract address.
DebtToken	➢ The token manager can update the name and symbol of the token. ➢ The token manager can update the stock token and exchange rate. ➢ The token manager can mint debt tokens to any arbitrary wallet. ➢ The stability pool manager address can transfer tokens from the sender wallet to the pool address.
HintHelpers	➢ The owner can update the sorted troves, trove manager, redemption operation, and price feed contract address.
LiquidationOperations	➢ The owner can update the trove manager, storage pool, price feed, token manager, stability pool manager, coll surplus pool contract address.
PriceFeed	➢ The owner can set the pyth and token manager contract address. ➢ The token manager can add new oracle IDs in the

	contract. ➢ The owner can update the alternative price feed contract address.
RedemptionOperations	➢ The owner can update the trove manager, storage pool, price feed, token manager, sorted troves, hint helpers contract address.
ReservePool	➢ The owner can update the stability pool manager address, price feed, and token manager contract address. ➢ The owner can update the relative stable cap value in the contract to any arbitrary number. ➢ The owner can update the gov reserve ca value to any arbitrary number in the contract.
SortedTroves	➢ The owner can update the trove manager, borrower operation, and redemption operation contract address.
StabilityPool	➢ The stability pool manager address can send the gains to the depositor wallet. ➢ The stability pool manager address withdraw all depositor's accumulated gains to depositor. ➢ The stability pool manager address can withdraw gains from the contract.
StabiilityPoolManager	➢ The owner can update the liquidation operations, price feed, storage pool, reserve pool, token manager contract address. ➢ The token manager can add the stability pools in the contract.
StakingOperations	➢ The owner can set the swap operations, token manager contract address. ➢ The token manager can set the rewards per second value. ➢ The token manager or swap operations contract address can set pool in the contract.
StoragePool	➢ The owner can set the borrower operations, trove manager, redemption operations, liquidation operations, stability pool and price feed contract address. ➢ The borrower operations, trove manager, stability pool manager, redemption operations, liquidation operations contract address can add, subtract, withdraw tokens from the contract.

SwapOperations	➢ The owner can create pair in the contract. ➢ The owner can set the swap base fees not more than 10%. ➢ The owner can update the swap gov fee to 100%.
SwapPair	➢ The swap operations can update the token0, token1, token manager, price feed, staking operation contract address.
TokenManager	➢ The owner can update the stability pool manager, staking operations, price feed, gov payout address in the contract. ➢ The owner can update ethe rewards per second value to any arbitrary amount in the staking operation contract address. ➢ The owner can set the staking alloc point in the staking operations contract address. ➢ The owner can enable/disable minting in the contract. ➢ The owner can disable or enable the debt minting. ➢ The owner can update ethe name and symbol of the debt tokens. ➢ The owner can update ethe next stock split value in the debt token address. ➢ The owner can set the stock exchange parameters in the debt token contract address. ➢ The owner can update the oracle ID in the price feed contract address. ➢ The owner can update the debt in the contract without the oracle ID. ➢ The owner can update the coll token with or without the oracle ID. ➢ The owner can update the coll token with supported collateral ratio.
TroveManager	➢ The owner can update the borrower, redemption operations, liquidation operation, storage pool, price feed sorted troves, token manager, reserve pool contract address. ➢ The owner can enable/disable liquidation. ➢ The owner can enable/disable redeeming. ➢ The owner can set the borrowing fee floor upto 100%.

- The owner can update ethe borrowing interest rate upto 100%.
- The borrower operations, redemption operation, liquidation operation address can set the trove status for the borrowers.
- The borrower operations, redemption operation, liquidation operation address can update ethe borrower's stake based on their latest collateral value.
- The borrower operations, redemption operation, liquidation operation address can remove the total stakes sum and their sakes to zero.
- The borrower operations, redemption operation, liquidation operation address can update the snapshots of system total stakes and collateral.
- The borrower operations, redemption operation, liquidation operation address can apply for the pending rewards.
- The borrower operations, redemption operation, liquidation operation address can update the trove reward snapshots.
- The borrower operations, redemption operation, liquidation operation address can increase/decrease trove call.
- The borrower operations, redemption operation, liquidation operation address can increase/decrease trove debt.

Recommendations

To avoid potential hacking risks, it is advisable for the client to manage the private key of the privileged account with care. Additionally, we recommend enhancing the security practices of centralized privileges or roles in the protocol through a decentralized mechanism or smart-contract-based accounts, such as multi-signature wallets.

Here are some suggestions of what the client can do:

- Consider using multi-signature wallets: Multi-signature wallets require multiple parties to sign off on a transaction before it can be executed, providing an extra layer of security e.g. Gnosis Safe
- Use of a timelock at least with a latency of e.g. 48-72 hours for awareness of privileged operations

- Introduce a DAO/Governance/Voting module to increase transparency and user involvement
- Consider Renouncing the ownership so that the owner cannot modify any state variables of the contract anymore. Make sure to set up everything before renouncing.

Audit Result

Critical Issues

No critical issues

High Issues

No high issues

Medium Issue

#1 | Missing 'require' check.

File	Severity	Location	Status
AlternativePriceFeed.sol	Medium	L137	Open

Description – The contract contains the functionality in which the price calculation is done which the help of the non-target USD and target weight and balance in the contract in this function the value of nontarget weight and balance should not be zero otherwise the functionality will fail and the price calculation will not be possible.

#2 | Missing 'isContract' check.

File	Severity	Location	Status
SwapPair.sol	Medium	L46-60	Open

Description — The contract admin can update any arbitrary address as the factory address, but this is not recommended. If the address is set to any arbitrary address, the contract's functionality will fail. There must be a check so that the address can only be set to a contract address.

#3 | Possible front-running.

File	Severity	Location	Status
RedemptionOperations.sol	Medium	L79-203	Open

Description — The function adjusts the collateral ratios (CR) of troves based on their current state and applies redemptions accordingly. If a large redemption is about to occur, it might lower the Total Collateral Ratio

(TCR) of the system or impact the collateral available for subsequent redemptions. A front-runner could preemptively redeem stablecoins before the TCR decreases or before the system enters Recovery Mode, where certain conditions might apply (e.g., no redemptions are allowed).

Remediation – Implementing fair ordering protocols, like batch auctions or Time-Bound Mempool Access, could prevent front-runners from gaining an advantage by observing pending transactions.

#4 | Missing threshold.

File	Severity	Location	Status
ReservePool.sol	Medium	L57-60, L62-65	Open
StakingOperations.sol	Medium	L57-67	Open

Description – The contract contains the functionality in which the owner can update any arbitrary value in the relatively stable cap or gov reserve cap, including zero, which is not recommended as this can fail the functionalities if the value is set to zero. Also, in the staking operations contract, the token manager can update any arbitrary value in the rewards per second value in the contract, which can set the rewards value to zero. There must be a check so that the rewards in the contract should not be zero.

Low Issue

#1 | Floating pragma solidity version.

File	Severity	Location	Status
All	Low	--	Open

Description – Adding the constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

#2 | Missing events arithmetic.

File	Severity	Location	Status
StabilityPoolManager.sol	Low	L188-199, L201-211, L213-220	Open
SwapOperations.sol	Low	L104-107, L113-116	Open
SwapPair.sol	Low	L46-60	Open

Description – It is recommended to emit all the critical parameter changes.

Informational Issue

#1 | Natspec documentation missing.

File	Severity	Location	Status
All	Informational	--	Open

Description – If you started to comment on your code, also comment on all other functions, variables, etc.

Legend for the Issue Status

Attribute or Symbol	Meaning
Open	The issue is not fixed by the project team.
Fixed	The issue is fixed by the project team.
Acknowledged(ACK)	The issue has been acknowledged or declared as part of business logic.

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY