CERTIK

Security Assessment

Milkomeda

May 11th, 2022

Table of Contents

Summary

This report has been prepared for Milkomeda to discover issues and vulnerabilities in the source code of the Milkomeda project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Milkomeda
Platform	Cardano
Language	Solidity
Codebase	https://github.com/dcSpark/milkomeda-validator/
Commit	1a8c6f20c73c7817c448c658f8bc502f6daa6776

Audit Summary

Delivery Date	May 11, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

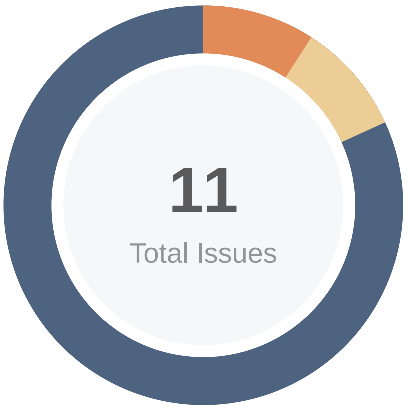
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0	0
● Major	1	0	0	0	0	0	1
● Medium	0	0	0	0	0	0	0
● Minor	1	0	0	0	0	0	1
● Informational	9	0	0	8	0	0	1
● Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
TMC	Types.sol	d82b7ff044a05ccdc2fc8f4bad956f6f2b196bf8a8bc39fc0c53c8eb76e688f9
STM	dev/SampleTokens.sol	67707d7f2923a18bd13e9d55acf7f9050581c7a9437625e90f2b038f7d29729b
MMC	Migrations.sol	36843b9bddd31153133949f23ce65cd0fa2d91cc5f0ac36298ba07d39de7fecd
PMC	proxy/Proxy.sol	8b3d0806382132c6396164a316339926fc9e32b105f1ae0a7045280416ee7d3f
SBM	SidechainBridge.sol	a98f2db68fdf1cbd07a9e22e1ccea657809e5c01d05795e9861b772efc1032e1
SBD	dev/SidechainBridgeDev.sol	9937b8b7ad1928473bad3ceffea65ef0b67dc254a32a52d3988d90ac14cbf253
SMC	State.sol	b2e17767b9f492e2673759267c154baef57b0a2b693655463beef9b1fce3854a
MMK	Multisig.sol	76712d9c4a5c3e8bc10be28de4c52e3bd378e98c28e4e34bfa9510613ef9856a
SBU	dev/SidechainBridgeUpgrade.sol	1bdb95fa4c508a13985f425a50bef89c4f2f2ca3b06a432720d38abf2e649c2f
TRM	TokenRegistry.sol	3a6b1d92b8c286668edd887ec2b9ad22b91e753df78755d0fe4dbde3d165436b

Findings



🟥 **Critical**	**0** (0.00%)
🟧 **Major**	**1** (9.09%)
🟨 **Medium**	**0** (0.00%)
🟨 **Minor**	**1** (9.09%)
🟦 **Informational**	**9** (81.82%)
🟩 **Discussion**	**0** (0.00%)

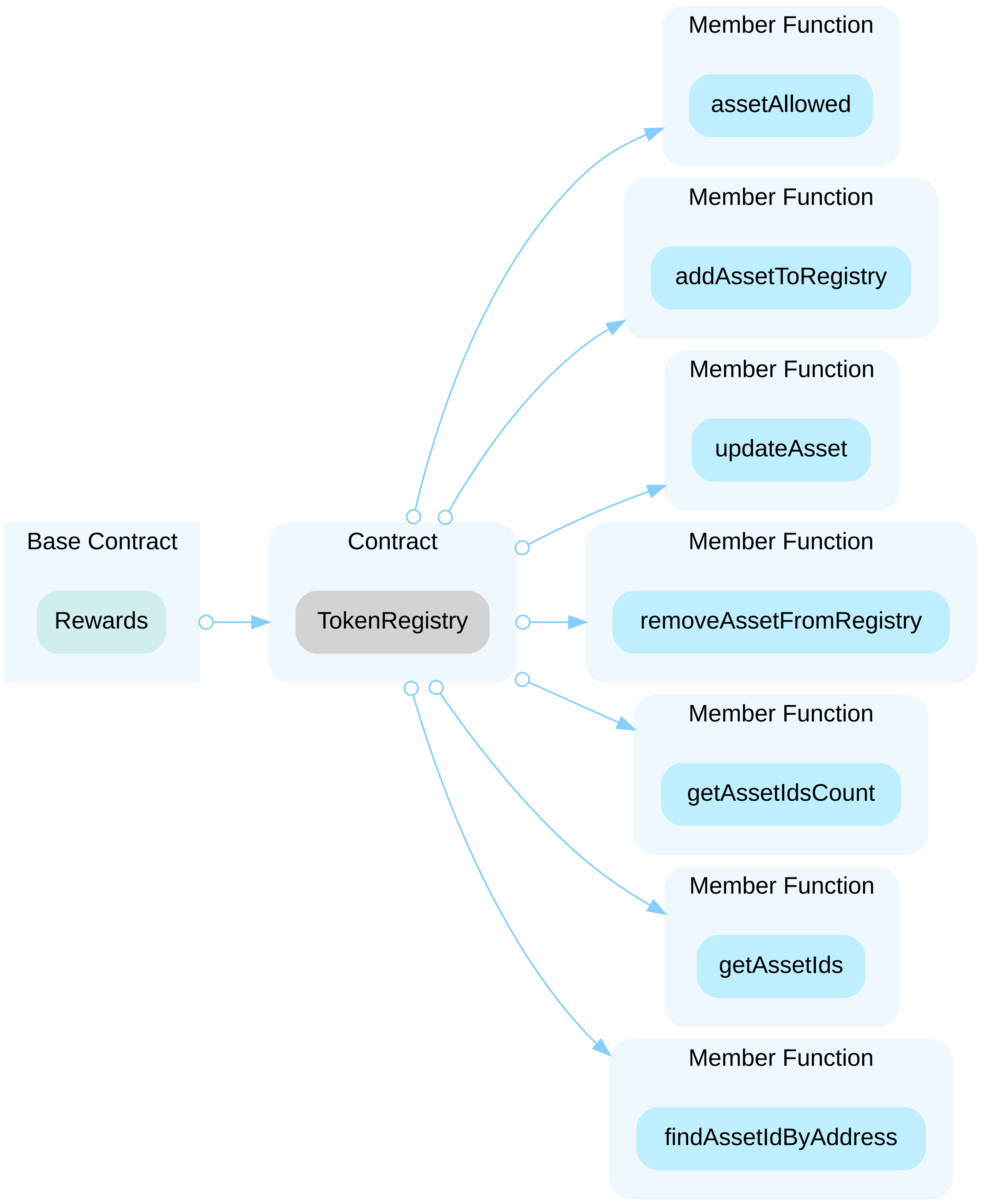
ID	Title	Category	Severity	Status
GLOBAL-01	Centralization Related Risks	**Centralization / Privilege**	● **Major**	✓ Resolved
MCK-01	Unlocked Compiler Version	Language Specific	● Informational	ⓘ Acknowledged
MCK-02	Improper Usage Of `public` And `external` Type	Gas Optimization	● Informational	ⓘ Acknowledged
MCK-03	Missing Input Validation	Volatile Code	● Informational	ⓘ Acknowledged
MCK-04	Testing Only Functions	Logical Issue	● Informational	✓ Resolved
MMK-01	Missing Handling The Case When Transaction Has Been Executed	Logical Issue	● Minor	✓ Resolved
SBM-01	Missing Emit Events	Coding Style	● Informational	ⓘ Acknowledged
SBM-02	Typo	Coding Style	● Informational	ⓘ Acknowledged
SBM-03	Cross-Chain Token Transfers	Volatile Code	● Informational	ⓘ Acknowledged
SBM-04	Intended Functionality For Validator	Volatile Code	● Informational	ⓘ Acknowledged
SBM-05	Potential Frontrun Initialization	Logical Issue	● Informational	ⓘ Acknowledged

GLOBAL-01 | Centralization Related Risks

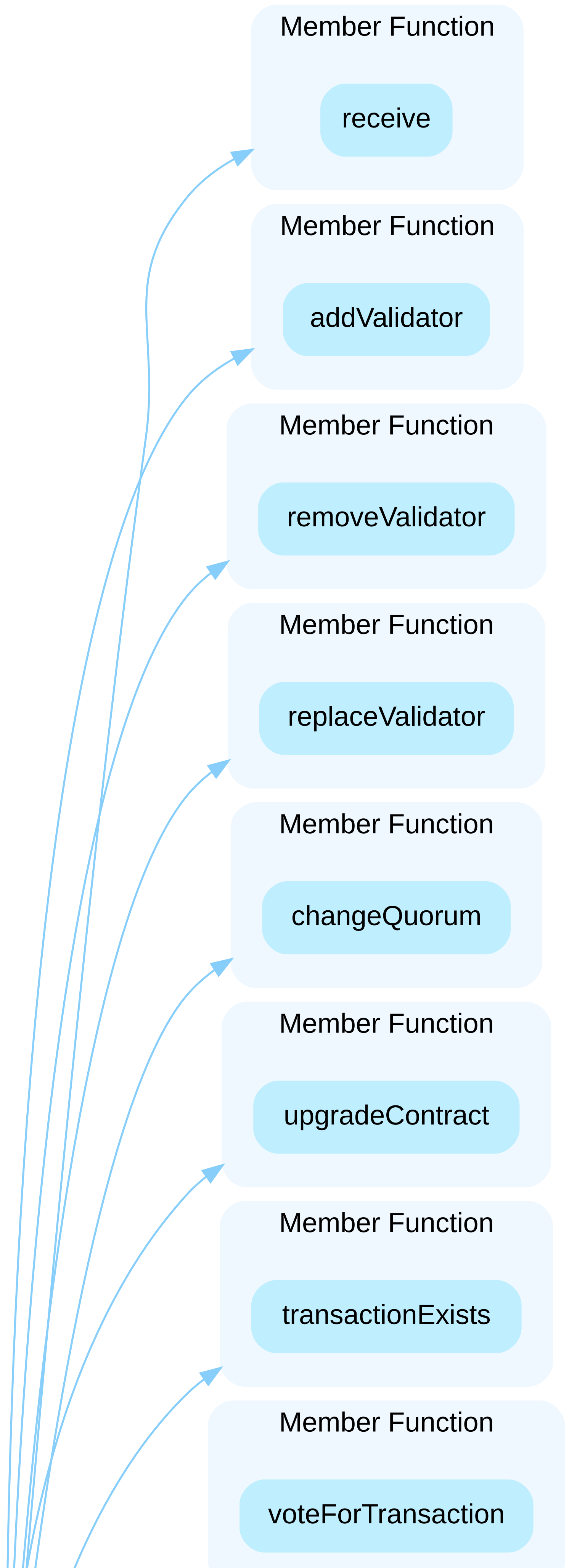
Category	Severity	Location	Status
Centralization / Privilege	● **Major**		Resolved

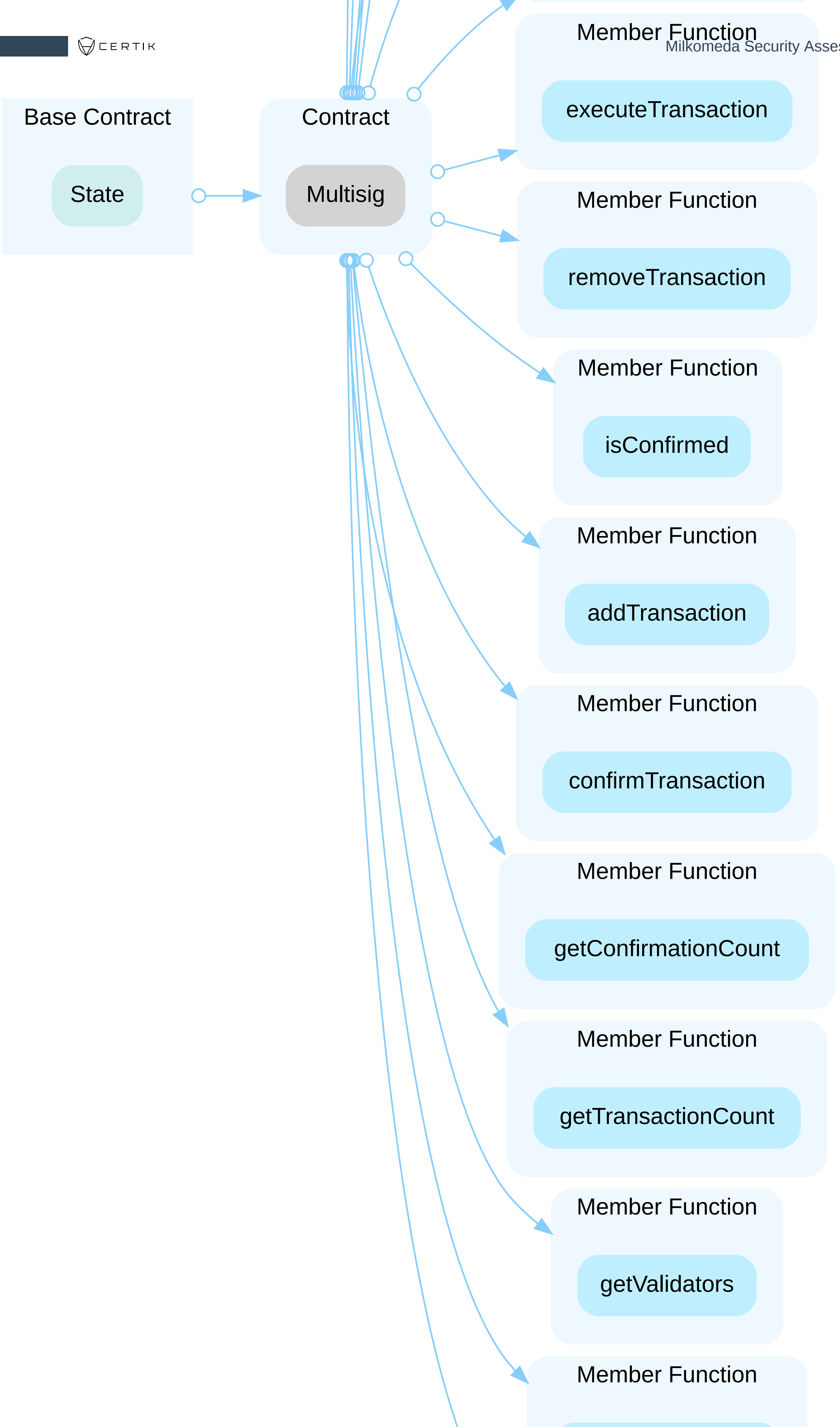
Description

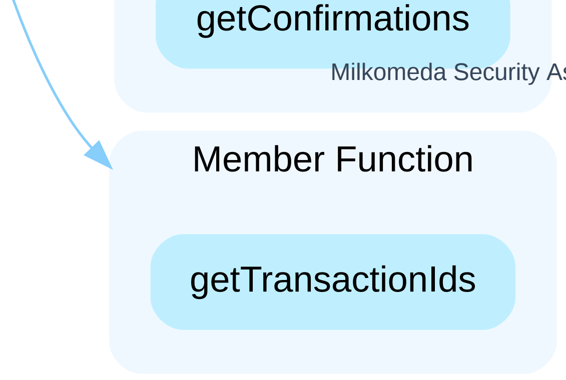
In the contract [TokenRegistry].sol the role `[onlyBridge]` has authority over the functions shown in the diagram below.



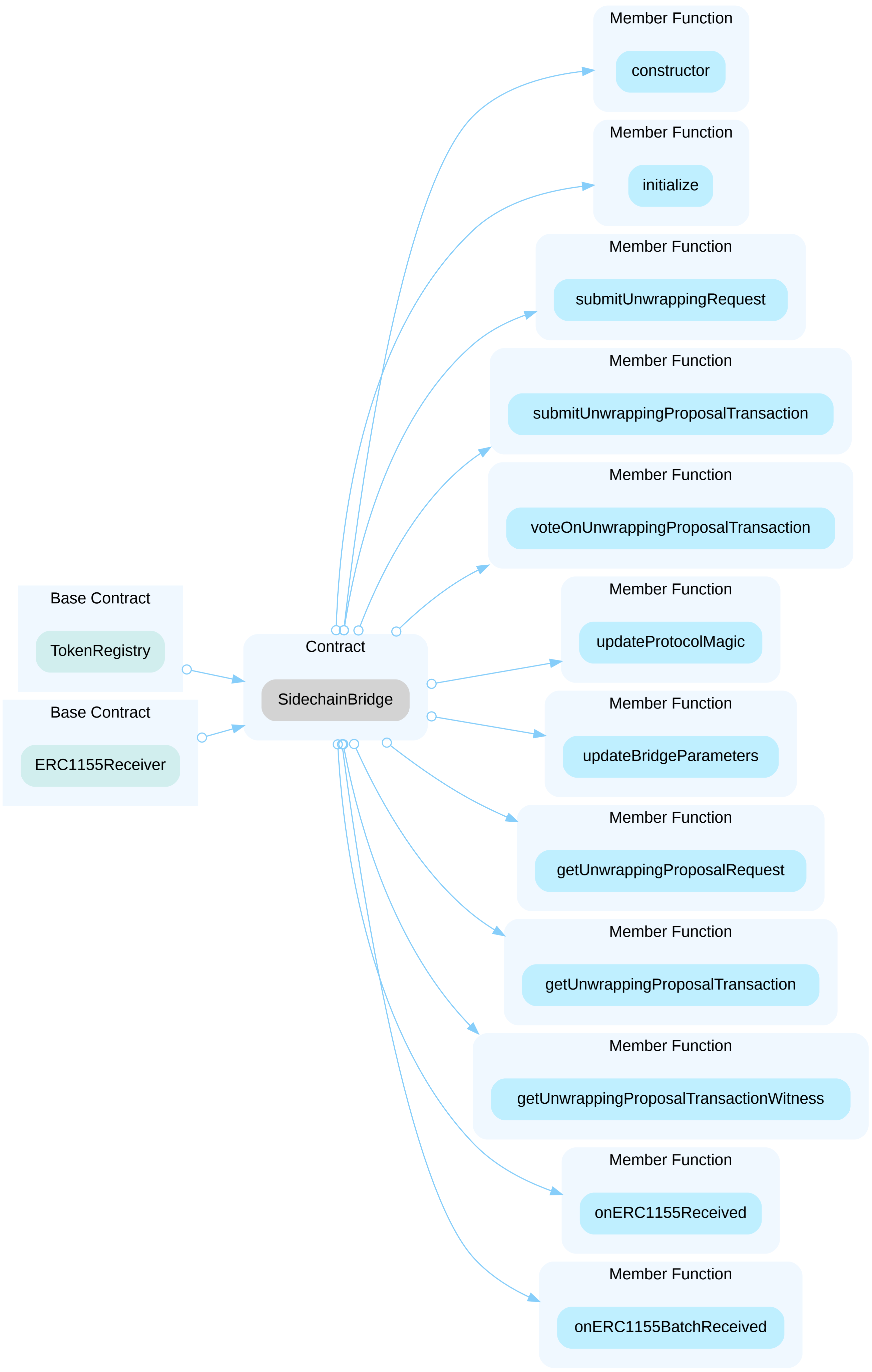
In the contract [Multisig].sol the role `[onlyBridge]` and `[validatorExists]` has authority over the functions shown in the diagram below.


Member Function
receive
Member Function
addValidator
Member Function
removeValidator
Member Function
replaceValidator
Member Function
changeQuorum
Member Function
upgradeContract
Member Function
transactionExists
Member Function
voteForTransaction


Base Contract
State
Contract
Multisig
Member Function
executeTransaction
Member Function
removeTransaction
Member Function
isConfirmed
Member Function
addTransaction
Member Function
confirmTransaction
Member Function
getConfirmationCount
Member Function
getTransactionCount
Member Function
getValidators
Member Function



In the contract [SidechainBridge.sol] the role `[validatorExists]` and `[onlyBridge]` has authority over the functions shown in the diagram below.


Member Function
constructor
Member Function
initialize
Member Function
submitUnwrappingRequest
Member Function
submitUnwrappingProposalTransaction
Member Function
voteOnUnwrappingProposalTransaction
Base Contract
TokenRegistry
Member Function
updateProtocolMagic
Contract
SidechainBridge
Base Contract
ERC1155Receiver
Member Function
updateBridgeParameters
Member Function
getUnwrappingProposalRequest
Member Function
getUnwrappingProposalTransaction
Member Function
getUnwrappingProposalTransactionWitness
Member Function
onERC1155Received
Member Function
onERC1155BatchReceived

Any compromise to the privileged accounts may allow a hacker to take advantage of this authority and update the sensitive settings and executive sensitive functions of the project.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR

- Remove the risky functionality.

Alleviation

`[Milkomeda]`: Please note that the authority of the `onlyBridge` modifier is the audited contract itself. This is a key part of its multi signature mechanism -- the contract will only ever become a caller if there were enough votes provided by validator (owner) majority.

Concerning the `validatorExists` modifier, it checks if `msg.sender` belongs to the current set of validators (owners). What this set is composed of however is again controlled by validator (owner) majority via multisig voting mechanism, i.e. functions protected by the `onlyBridge` modifier. Thus here there is no centralization as well.

In short, there already is decentralization via multi signatures, the second modifier benefits from it and the first one is part of its implementation.

Proxy: https://explorer-mainnet-cardano-evm.c1.milkomeda.com/address/0x000000000000000000000000000000000000BbBB/transactions

Logic: https://explorer-mainnet-cardano-evm.c1.milkomeda.com/address/0x0000000000000000000000000000000000001111/internal-transactions

MCK-01 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	● Informational	Migrations.sol: 2; Multisig.sol: 2; proxy/Proxy.sol: 2; Rewards.sol: 2; SidechainBridge.sol: 2; TokenRegistry.sol: 2, 2; State.sol: 2; Types.sol: 2	ⓘ Acknowledged

Description

The contract has unlocked compiler version. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

We advise that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.8.7` the contract should contain the following line:

```
pragma solidity 0.8.7;
```

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

MCK-02 | Improper Usage Of `public` And `external` Type

Category	Severity	Location	Status
Gas Optimization	● Informational	TokenRegistry.sol: 22, 35, 46~56; Multisig.sol: 148~150, 226~238; SidechainBridge.sol: 155, 234, 264, 272, 322~330, 332~340	ⓘ Acknowledged

Description

`public` functions that are never called by the contract could be declared as `external`. `external` functions are more efficient than `public` functions.

Recommendation

Consider using the external attribute for public functions that are never called within the contract.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

MCK-03 | Missing Input Validation

Category	Severity	Location	Status
Volatile Code	● Informational	SidechainBridge.sol: 264, 272; Multisig.sol: 148	ⓘ Acknowledged

Description

There is no validation to check whether the inputs are less than a certain value, changed, or zero address.

Recommendation

We advise adding the check for the passed-in values to prevent unexpected error.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

MCK-04 | Testing Only Functions

Category	Severity	Location	Status
Logical Issue	● Informational	dev/SidechainBridgeDev.sol: 6; dev/SidechainBridgeUpgrade.sol: 6	✓ Resolved

Description

Per the comments in the `SidechainBridgeDev.sol` and `SidechainBridgeUpgrade.sol`, the functions in these two contracts are for testing purposes only. The team should not use these functions in the production environment to avoid any misconfiguration.

Recommendation

Consider to exclude these contracts from the scope of the deployment script

Alleviation

`[Milkomeda]`: Indeed. They are not used in production.

MMK-01 | Missing Handling The Case When Transaction Has Been Executed

Category	Severity	Location	Status
Logical Issue	● Minor	Multisig.sol: 171~194	✓ Resolved

Description

In the function `voteForTransaction()`, the case that a specific transaction exists and has been executed, is not handled properly. This will miss catching the potential execution case and also may waste gas in this case.

Recommendation

Consider to add logic to handle the case:

```
if (!transactions[transactionId].executed){
  confirmTransaction(transactionId);
}else{
  //handle the case when the transaction has been executed, like revert or emit log
}
```

Alleviation

`[Milkomeda]`: A transaction will be executed as soon as the quorum is reached. However it is possible that other validators votes are yet to be registered by the smart contract. I.e. all the validator will check independently from each other if an action can be done and will all vote as soon as possible to execute a transaction. If we were to throw an error on votes that are not necessary (but not invalid) we would have up to validators - quorum error reported to all the nodes this will create a non necessary noise. Once the transaction has been executed all the validators are notified by an event the transaction was executed.

SBM-01 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	● Informational	SidechainBridge.sol: 264, 272	ⓘ Acknowledged

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to users.

- `setCompleted()`

Recommendation

Consider adding events for sensitive actions and emit them in the function.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

SBM-02 | Typo

Category	Severity	Location	Status
Coding Style	● Informational	SidechainBridge.sol: 19	ⓘ Acknowledged

Description

```
///    On an unwrapping request, if the bridge is able to obtain the signalled
```

```
/// vote for the transaction is added. If any argument differs from its
```

Recommendation

Current spelling would be:

```
///    On an unwrapping request, if the bridge is able to obtain the signaled
```

Could be written better as:

```
/// vote for the transaction added. If any argument differs from its
```

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

SBM-03 | Cross-Chain Token Transfers

Category	Severity	Location	Status
Volatile Code	● Informational	SidechainBridge.sol	ⓘ Acknowledged

Description

Although not explicitly declared, the contract `SidechainBridge` serves as a tool to help users perform cross-chain token transfer. The cross-chain token transfer may depend on third party cross-chain protocol or off-chain scripts. The scope of the audit treats these 3rd party entities as black boxes and assume their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, etc.

Recommendation

We understand that the business logic of `SidechainBridge` requires interaction with third party cross-chain protocol or off-chain scripts. We encourage the team to ensure their functional correctness and constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

SBM-04 | Intended Functionality For Validator

Category	Severity	Location	Status
Volatile Code	● Informational	SidechainBridge.sol: 121	ⓘ Acknowledged

Description

Contract can be initialized with one validator. This would seem to defeat the purpose of a decentralized consensus mechanism and give the power to vote to push or stop any transactions through the bridge.

Recommendation

We want to make sure this functionality is intended.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

SBM-05 | Potential Frontrun Initialization

Category	Severity	Location	Status
Logical Issue	● Informational	SidechainBridge.sol: 108	ⓘ Acknowledged

Description

The function `initialize()` can be called by any external caller if it has never been called previously. This can be taken advantage by attacker through frontrun and call the `initialize()` function before the contract deployer

Recommendation

We advise the team to consider using script or a smart contract to deploy the project and call `initialize()` function making sure the deployment and `initialize()` invocation happened in the single one transaction.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK’s prior written consent in each instance.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.