Three Sigma

Code Audit

M^0 [Labs]

M^0 Labs

Coordination layer for permissioned institutional actors to generate $M

Disclaimer

Code Audit

M^0 Coordination layer for permissioned institutional actors to generate $M

Disclaimer

The ensuing audit offers no assertions or assurances about the code's security. It cannot be deemed an adequate judgment of the contract's correctness on its own. The authors of this audit present it solely as an informational exercise, reporting the thorough research involved in the secure development of the intended contracts, and make no material claims or guarantees regarding the contract's post-deployment operation. The authors of this report disclaim all liability for all kinds of potential consequences of the contract's deployment or use. Due to the possibility of human error occurring during the code's manual review process, we advise the client team to commission several independent audits in addition to a public bug bounty program.

Table of Contents

Table of Contents

Summary

Code Audit

M^0 Coordination layer for permissioned institutional actors to generate $M

Summary

Three Sigma audited M^0 in a 3 day engagement. The audit was conducted from 17-07-2024 to 19-07-2024.

Protocol Description

A neutral value transmission framework able to permissionlessly mint currencies under decentralized governance. The purpose of $M is to become a superior building block for value representation, by combining the convenience of digital money with the risk profile of physical cash.

Scope

Code Audit

M^0 Coordination layer for permissioned institutional actors to generate $M

Scope

Filepath	nSLOC
src/libs/IndexingMath.sol	46
src/Migratable.sol	20
src/Proxy.sol	31
src/WrappedMToken.sol	239
Total	**336**

Methodology

Code Audit

M^0 Coordination layer for permissioned institutional actors to generate $M

Methodology

To begin, we reasoned meticulously about the contract's business logic, checking security-critical features to ensure that there were no gaps in the business logic and/or inconsistencies between the aforementioned logic and the implementation. Second, we thoroughly examined the code for known security flaws and attack vectors. Finally, we discussed the most catastrophic situations with the team and reasoned backwards to ensure they are not reachable in any unintentional form.

Taxonomy

In this audit we report our findings using as a guideline Immunefi's vulnerability taxonomy, which can be found at immunefi.com/severity-updated/. The final classification takes into account the severity, according to the previous link, and likelihood of the exploit. The following table summarizes the general expected classification according to severity and likelihood; however, each issue will be evaluated on a case-by-case basis and may not strictly follow it.

Severity / Likelihood	LOW	MEDIUM	HIGH
NONE	None		
LOW	Low		
MEDIUM	Low	Medium	Medium
HIGH	Medium	High	High
CRITICAL	High	Critical	Critical

Project Dashboard

Code Audit

M^0 Coordination layer for permissioned institutional actors to generate $M

Project Dashboard

Application Summary

Name	M^0
Commit	0dd7b4553009e3d87298a2f9e6aec2a89ba308ad
Fix Commit	55896c2f37a13a39fae933d52a2b6687ff4c9408
Language	Solidity
Platform	Ethereum

Engagement Summary

Timeline	17-07-2024 to 19-07-2024
№ of Auditors	1
Review Time	3 days

Vulnerability Summary

Issue Classification	Found	Addressed	Acknowledged
Critical	0	0	0
High	0	0	0
Medium	0	0	0
Low	2	1	1

None	2	0	2

Category Breakdown

Suggestion	4
Documentation	0
Bug	0
Optimization	0
Good Code Practices	0

Code Maturity Evaluation

Code Audit

M^0 Coordination layer for permissioned institutional actors to generate $M

Code Maturity Evaluation

Code Maturity Evaluation Guidelines

Category	Evaluation
Access Controls	The use of robust access controls to handle identification and authorization and to ensure safe interactions with the system.
Arithmetic	The proper use of mathematical operations and semantics.
Centralization	The presence of a decentralized governance structure for mitigating insider threats and managing risks posed by contract upgrades
Code Stability	The extent to which the code was altered during the audit.
Upgradeability	The presence of parameterizations of the system that allow modifications after deployment.
Function Composition	The functions are generally small and have clear purposes.
Front-Running	The system's resistance to front-running attacks.
Monitoring	All operations that change the state of the system emit events, making it simple to monitor the state of the system. These events need to be correctly emitted.
Specification	The presence of comprehensive and readable codebase documentation.
Testing and Verification	The presence of robust testing procedures (e.g., unit tests, integration tests, and verification methods) and sufficient test coverage.

Code Maturity Evaluation Results

Category	Evaluation
Access Controls	Satisfactory. All access control is correctly implemented.
Arithmetic	Moderate. Some rounding errors were found.
Centralization	Satisfactory. No significant points of centralization were found.
Code Stability	Satisfactory. The code was stable throughout the audit.
Upgradeability	Satisfactory. The contracts are upgradeable.
Function Composition	Satisfactory. The code was correctly split into helper functions.
Front-Running	Satisfactory. No front-running issues were found.
Monitoring	Satisfactory. Most events are emitted.
Specification	Satisfactory. The code follows the specifications.
Testing and Verification	Satisfactory. The codebase implements unit and fuzz tests.

Findings

Code Audit

M^0 Coordination layer for permissioned institutional actors to generate $M

Findings

3S-M^0-L01

Anyone can stop earning $M directly through **`MToken::stopEarning(address account_)`** increasing the **`WrappedMToken`** balance without backing

Id	3S-M^0-L01
Classification	Low
Severity	Medium
Likelihood	Low
Category	Suggestion
Status	Acknowledged

Description

`MToken` allows an account to stop earning if it is not approved anymore in the **`Registrar`**. For this reason, when the **`WrappedMToken`** stops being approved, it must call stop earning in the **`MToken`** and store the last **`index`**, such that the correct balances can be calculated. However, **`MToken::stopEarning(address account_)`** is permissionless and enables sending the **`WrappedMToken`** address, stopping **`MToken`** from being accrued. However, as it is not called through **`WrappedMToken::disableEarning()`**, the **`_lastDisableEarningIndex()`** will not be immediately recorded, accruing artificial **`WrappedMToken`** balance that is not actually backed. Thus, users will be able to withdraw more **`MToken`** than what **`WrappedMToken`** holds, potentially leading to the last users not being able to withdraw, depending on the index mismatch.

Recommendation

The governance call to stop the **`WrappedMToken`** from being approved in the **`Registrar`** must be called atomically with a call to **`WrappedMToken::disableEarning()`**, such that the last index is correctly recorded.

3S-M^0-L02

`_subtractTotalEarningSupply()` could not be in an unchecked block to prevent rounding errors from wrapping the supply

Id	3S-M^0-L02
Classification	Low
Severity	Low
Likelihood	Low
Category	Suggestion
Status	Addressed in #d6f4404.

Description

On **`WrappedMToken::_addEarningAmount()`**, the principal of an account is set as **`IndexingMath.getPrincipalAmountRoundedDown(balance_ + amount_, index_)`**, but the added principal to **`_principalOfTotalEarningSupply`** is **`IndexingMath.getPrincipalAmountRoundedDown(amount_, currentIndex_)`**, indicating that there will be a mismatch between the sum of account principal and **`_principalOfTotalEarningSupply`**. Additionally, the added amount to **`totalEarningSupply()`** can further round down in **`WrappedMToken::_setTotalEarningSupply()`** when calculating the index. In **`WrappedMToken_::subtractEarningAmount()`**, the same happens as the balance of the account is updated with **`balance_ - amount`** but the subtract principal is calculated on **`amount`** only, and the index calculation can again round down. Thus, we can expect the sum of the individual accounts principal and balance to deviate slightly (can be considered dust and dealt with using **`excess()`**, as long as the balance of **`MToken`** is bigger).

Recommendation

In **`WrappedMToken::_subtractTotalEarningSupply()`**, remove the **`unchecked`** block as the mismatch can cause this operation to wrap around, leading to inflated total supplies.

3S-M^0-N01

Re-enabling, although not possible at the moment, will make a jump in the index, not accompanied by the $M balance increase

Id	3S-M^0-N01
Classification	None
Category	Suggestion
Status	Acknowledged

Description

Firstly this bug is not possible at the moment because re-enabling is disabled, see **`WrappedMToken::enableEarning()`**. However, if it is allowed, when it is enabled again, it will use the current index from the **`MToken`**, which will be bigger than the past stored index in **`WrappedMToken`** while it was disabled. This will make a jump in everyone's **`WrappedMToken`** balance, but the actual **`MToken`** holdings will not match it. Thus, users would get instant profit while the last ones would not be able to withdraw.

Recommendation

A simple fix is not trivial. Perhaps when the new index is fetched, a multiplier can be applied that discounts the previous stored index.

3S-M^0-N02

`WrappedMToken::_lastDisableEarningIndex()` assumes the index will be in position 1 when disabled, which may not be true in the future

Id	3S-M^0-N02
Classification	None
Category	Suggestion
Status	Acknowledged

Description

`WrappedMToken::_lastDisableEarningIndex()` fetches index 1 from **`_enableDisableEarningIndices`** to get the last stored index before disabling. However, this is only true if **`WrappedMToken`** is disabled once, but if the current code to stop re-enabling earning is removed, it will lead to a wrong index.

Recommendation

The correct index is **`_enableDisableEarningIndices.length - 1`**. If for gas saving purposes 1 is kept, a comment should be added to ensure it is changed later in case the re-enabling is allowed in the future.