Three Sigma

Code Audit

Syrup Lending Protocol

Disclaimer

Code Audit

Syrup Lending Protocol

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

Code Audit

Syrup Lending Protocol

Table of Contents

Summary

Code Audit

Syrup Lending Protocol

Summary

Three Sigma audited Syrup in a 5 days engagement. The audit was conducted from 19-08-2024 to 23-08-2024.

Protocol Description

The Syrup platform, built by Maple Labs, enables users permissionless access to secured, institutional lending for the first time. By depositing USDC into the platform, users receive LP tokens (syrupUSDC) and begin earning yield immediately. All of the yield generated by Syrup is sourced from secured loans to the largest institutions in crypto, fully collateralized with digital assets.

Scope

Code Audit

Syrup Lending Protocol

Scope

Filepath
mpl-migration/Migrator.sol
syrup-utils/SyrupDrip.sol
syrup-utils/SyrupUserActions.sol
syrup-utils/MPLUserActions.sol
fixed-term-loan/MapleLoan.sol
open-term-loan/MapleLoan.sol

Assumptions

The scope of the audit was carefully defined to include the contracts at the lowest level of the inheritance hierarchy, as these are the ones that will be deployed to the mainnet. The libraries used in the implementation of these contracts are internal contracts that have been previously audited and shown to be secure. Additionally, the Balancer and Makerdao codebases are considered to be safe.

Methodology

Code Audit

Syrup Lending Protocol

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

Syrup Lending Protocol

Project Dashboard

Application Summary

Name	Syrup
Commit	**syrup-utils** 52d7d2252193ce24edea3e983ed3016557bac7d6 **open-term-loan** f71cc276f88a98cccd7b72ce8333b0f7477f2c87 **fixed-term-loan** 622c3b5d14799bfca02f9f59dd743fc13c4a9cb4 **mpl-migration** 3d49ab895cf4a13d26d3331a004aacb7858d9877
Language	Solidity
Platform	Ethereum

Engagement Summary

Timeline	19-08-2024 to 23-08-2024
Nº of Auditors	2
Review Time	5 days

Vulnerability Summary

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

None	4	2	2

Category Breakdown

Suggestion	1
Documentation	0
Bug	0
Optimization	2
Good Code Practices	2

Code Maturity Evaluation

Code Audit

Syrup Lending Protocol

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. The codebase has a strong access control mechanism.
Arithmetic	Satisfactory. The codebase uses Solidity version >0.8.0 as well as takes the correct measures in rounding the results of arithmetic operations.
Centralization	Weak. The Syrup protocol has significant privileges over the codebase.
Code Stability	Satisfactory. The code was stable during the audit.
Upgradeability	Satisfactory. Most contracts are upgradeable except for the stateless ones.
Function Composition	Satisfactory. Certain components are similar, and the codebase would benefit from increased code reuse.
Front-Running	Satisfactory. No front-running issues were found.
Monitoring	Satisfactory. Events are correctly emitted and the Syrup Router keeps track of on chain activity.
Specification	Satisfactory. In-depth and well structured high-level specification as well as codebase documentation.
Testing and Verification	Satisfactory. Extensive test code coverage as well as usage of tools and different test methods.

Findings

Code Audit

Syrup Lending Protocol

Findings

3S-Sy-L01

Deadline parameter in **`SyrupUserActions::_swapViaBalancer()`** could be set to minimize MEV

Id	3S-Sy-L01
Classification	Low
Severity	Low
Likelihood	Medium
Category	Suggestion
Status	Addressed in #4614e56

Description

`SyrupUserActions` sets a minimum amount out limit to prevent excessive MEV. However, as the deadline parameter is set to **`block.timestamp`** there is still some room for MEV.

Recommendation

Send a deadline parameter instead of hardcoding **`0`** to further mitigate MEV.

3S-Sy-N01

Extra spaces found in some instances of the codebase

Id	3S-Sy-N01
Classification	None
Category	Good Code Practices
Status	Addressed in #e9e7dff.

Description

Some extra spaces were found in the codebase which could be fixed for increased readability.

Recommendation

`bytes32 public immutable  override poolId;` in `SyrupUserActions`.

`// 4. If asset out is USDC, swap DAI  to USDC` in `SyrupUserActions::_swap()`.

3S-Sy-N02

Duplicated slippage check in **`SyrupUserActions`**

Id	3S-Sy-N02
Classification	None
Category	Optimization
Status	Addressed in #50438b9

Description

`SyrupUserActions::_swap()` checks the slippage at the end **`require(amountOut_ >= minAmountOut_, "SUA:S:INSUFFICIENT_AMOUNT_OUT");`**. When the asset out is **`USDC`**, it double checks the slippage in **`SyrupUserActions::_swapDaiForUsdc()`** as **`require(usdcOut_ >= minUsdcOut_, "SUA:SDU:INSUFFICIENT_AMOUNT_OUT");`**.

Recommendation

There is no need to check the slippage in **`SyrupUserActions::_swapDaiForUsdc()`**.

3S-Sy-N03

Duplicated **`_permit()`** function

Id	3S-Sy-N03
Classification	None
Category	Good Code Practices
Status	Acknowledged

Description

The **`_permit()`** function, utilized in the **`MplUserActions`**, **`SyrupRouter`**, and **`SyrupUserActions`** contracts, is duplicated across them.

Recommendation

To enhance code reusability and maintainability, this function can be relocated to the **`utils`** folder.

3S-Sy-N04

Immutable variables are emitted in events

Id	3S-Sy-N04
Classification	None
Category	Optimization
Status	Acknowledged

Description

The following immutable variables are unnecessarily included in event emissions:

- **`SyrupUserActions::_swap()`** - **`syrupUsdc`** in **`Swap`** event.

- **`MplUserActions::_redeemAndMigrateAndStake()`** - **`xmpl`** in **`RedeemedAndMigratedAndStaked`** event.

Recommendation

Immutable variables can not be changed, unless upgraded, so an event may not be necessary.