Multiplier-Finance

Security Assessment

January 25th, 2021

For :
Multiplier-Finance

By :
Alex Papageorgiou @ CertiK
alex.papageorgiou@certik.org

Adrian Hetman @ CertiK
adrian.hetman@certik.org

Disclaimer

CertiK 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 or project that contracts CertiK to perform a security review.

CertiK Reports 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 technologies proprietors, business, business model or legal compliance.

CertiK Reports 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.

CertiK 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. 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.

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has indeed completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.

Overview

Project Summary

Project Name	Multiplier-Finance
Description	N/A
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commits	1. 7da67c85d6d89a1025753df4c71a99425a6b502f 2. 9e440710d85b2ee1166106d4f9d7a36d4eaf3f0b

Audit Summary

Delivery Date	January 25th, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	2
Timeline	January 12th, 2021 - January 25th, 2021

Vulnerability Summary

Total Issues	18
Total Critical	0
Total Major	0
Total Medium	0
Total Minor	5
Total Informational	13

Executive Summary

We were tasked with auditing Multiplier Finance, specially their changes made to the lending and staking mechanisms.

The project is based on the AAVE protocol modules, such as the lending and safety modules for staking. Every contract, apart from one, was forked from the GitHub AAVE source code. Liquidation manager code within the linked repository was taken from AAVE's live, deployed version.
The team provided documentation which helped with understanding the changes. Nothing major was found, however we did advise the team to implement certain additional edge-test cases such as transfers to self to accommodate for the new rewarding mechanisms introduced.

Code changes are documented and closely conform to the coding style and standards of the original AAVE codebase.

Over the course of the audit, we examined the delta between the AAVE and the Multiplier Finance code in great detail, ensuring that the peculiar programming paradigms within AAVE's codebase were conformed to by the changes introduced in the linked repository.

To properly gauge the effect of the changes introduced by the Multiplier Finance team to the codebase, we also examined that the interfaces and inherent requirements of the various modules of the AAVE codebase new code interacts with are properly followed to ensure expected behaviour of the overall system.

We iterated over the new files using our suite of static analysis tools and additionally inspected the test coverage of the new repository to ensure edge cases that may introduce a vulnerability within the new code are accommodated for. To this end, we contacted the Multiplier Finance team wherever applicable and informed them of ways they can expand their test suite coverage.

All changes that were carried out in the AAVE codebase bear close resemblence to the coding style of the AAVE codebase itself, ensuring an easily-comprehensible code adjustment that also innately follows the style guidelines of AAVE as well as its functional side effects, such as the Checks-Effects-Interactions pattern.

The following files contained minimal changes mostly related to configurations necessitated by the new functionality introduced. Additionally, some of these contracts were simplified by omitting functionality that was deemed unnecessary in the new version:

- lending/contracts/configuration/LendingPoolAddressesProvider.sol
- lending/contracts/configuration/LendingPoolParametersProvider.sol
- lending/contracts/fees/FeeProvider.sol
- lending/contracts/lendingpool/LendingPool.sol
- lending/contracts/lendingpool/LendingPoolCore.sol
- lending/contracts/lendingpool/LendingPoolLiquidationManager.sol
- lending/contracts/tokenization/MToken.sol
- lending/contracts/libraries/CoreLibrary.sol
- staking/contracts/stake/StakedToken.sol

The `LendingPool.sol` and `LendingPoolCore.sol` contracts were adjusted to support the BNB token, as Multiplier Finance's AAVE fork is meant to be deployed on Binance, and introduced a new restriction on borrowing whereby up to 99% of the available liquidity is able to be borrowed rather than 100%. With regards to novel functionality, a new rewarding mechanism was

introduced that replaces the protocol's fees and redistributes them to the users. To this end, we would like to note that the new code introduced beyond L1169 of the `LendingPool.sol` file conducts expensive iterative loops over the reserves of the protocol, thus prohibiting a large amount of assets being added as reserves to it. The reward system introduced allows anyone to update the rewards of a specified address, however it solely allows the owner of the rewards to claim them via the corresponding introduced methods.

The `LendingPoolLiquidationManager.sol` was adjusted to accommodate for different-decimal assets in its `calculateAvailableCollateralToLiquidate` function to ensure support for any-decimal currency exists for the said function. Additionally, certain segments were adjusted to conform to the adaptations detailed above in `LendingPool` and `LendingPoolCore`.

Within the `CoreLibrary.sol`, a function was introduced that aids in the newly introduced check of the Lending Pools to not exceed 99% of the available liquidity in borrows.

The `StakedToken.sol` implementation was adjusted to update the Governance-based fee reward mechanisms and accommodate for them prior to any adjustment in the users' balances, however, we would like to note that the implementation needs to re-invoke the `updateGovernanceStakingRewards` depending on the type of token set to be the `STAKED_TOKEN` as the reward mechanisms could rely on it.

The remaining 2 contracts contained solely configurational changes for the new reward mechanisms.

Two new contracts have also been introduced to the codebase, namely:

- lending/contracts/rewards/RewardsManager.sol
- lending/contracts/vaults/RewardVault.sol

The vault acts as a deposit address for rewards that are meant to be redeemed at a certain point in the future by users, whereas the reward manager contains the core logic for maintaining and accumulating user rewards based on the fees generated by the system. We would like to state that the reward manager contains a lot of strict `require` checks that ensure a specified `reserve` has been added to the set of reward pools on the contract, thus potentially breaking the full breadth of features the forked AAVE codebase provides if a `reserve` hasn't been added to the reward pools in time. For this purpose, we advise that whenever a reserve is added the reward manager is automatically informed instead of iterating over all reserves on each insertion.

Overall, the codebase can be considered to be of a high standard and no medium severity and above issues have been identified in the codebase.

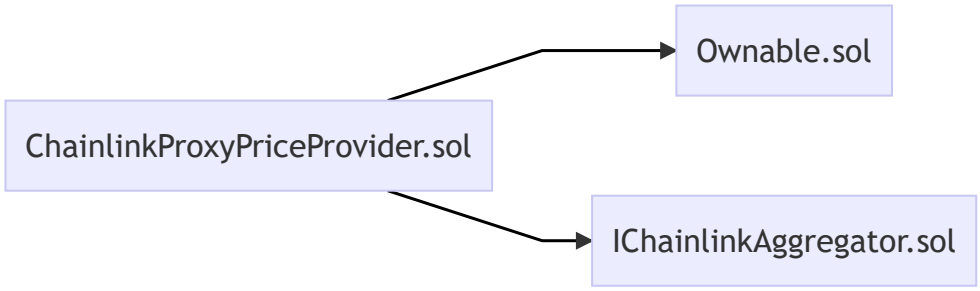
Files In Scope

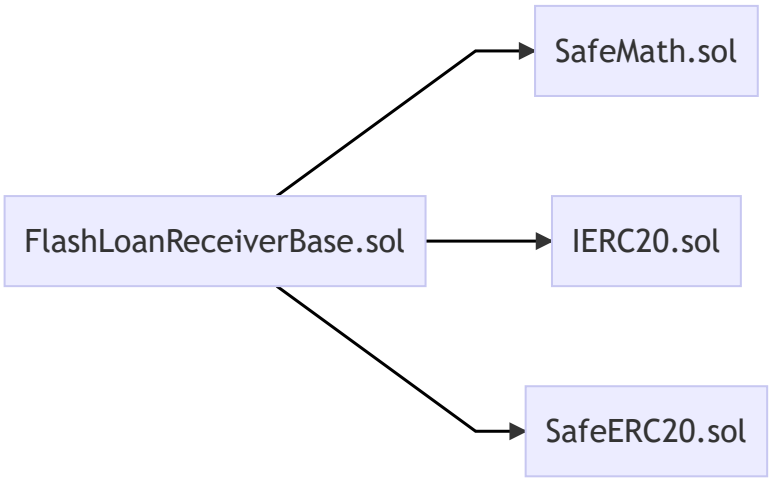
ID	Contract	Location
ADD	Address.sol	staking/contracts/lib/Address.sol
ASE	AddressStorage.sol	lending/contracts/configuration/AddressStorage.sol
AUP	AdminUpgradeabilityProxy.sol	lending/contracts/libraries/openzeppelin-upgradeability/AdminUpgradeabilityProxy.sol
BAL	BscAddressLib.sol	lending/contracts/libraries/BscAddressLib.sol
BUP	BaseUpgradeabilityProxy.sol	lending/contracts/libraries/openzeppelin-upgradeability/BaseUpgradeabilityProxy.sol
STA	BaseUpgradeabilityProxy.sol	staking/contracts/lib/BaseUpgradeabilityProxy.sol
BAU	BaseAdminUpgradeabilityProxy.sol	staking/contracts/lib/BaseAdminUpgradeabilityProxy.sol
BAP	BaseAdminUpgradeabilityProxy.sol	lending/contracts/libraries/openzeppelin-upgradeability/BaseAdminUpgradeabilityProxy.sol
CON	Context.sol	staking/contracts/lib/Context.sol
CLY	CoreLibrary.sol	lending/contracts/libraries/CoreLibrary.sol
CPP	ChainlinkProxyPriceProvider.sol	lending/contracts/misc/ChainlinkProxyPriceProvider.sol
DTN	DeedToken.sol	converter/bsc/contracts/DeedToken.sol
DTS	DistributionTypes.sol	staking/contracts/lib/DistributionTypes.sol
DRI	DefaultReserveInterestRateStrategy.sol	lending/contracts/lendingpool/DefaultReserveInterestRateStrategy.sol
ERC	ERC20.sol	staking/contracts/lib/ERC20.sol
ERW	ERC20WithSnapshot.sol	staking/contracts/lib/ERC20WithSnapshot.sol
FPR	FeeProvider.sol	lending/contracts/fees/FeeProvider.sol
FLR	FlashLoanReceiverBase.sol	lending/contracts/flashloan/base/FlashLoanReceiverBase.sol
IER	IERC20.sol	staking/contracts/interfaces/IERC20.sol
IBT	IBToken.sol	staking/contracts/interfaces/IBToken.sol
ISM	IStakedbMXX.sol	staking/contracts/interfaces/IStakedbMXX.sol
IFP	IFeeProvider.sol	lending/contracts/interfaces/IFeeProvider.sol
ILP	ILendingPool.sol	staking/contracts/interfaces/ILendingPool.sol
IPO	IPriceOracle.sol	lending/contracts/interfaces/IPriceOracle.sol
IRV	IRewardVault.sol	lending/contracts/interfaces/IRewardVault.sol
ITH	ITransferHook.sol	staking/contracts/interfaces/ITransferHook.sol
INI	Initializable.sol	lending/contracts/libraries/openzeppelin-upgradeability/Initializable.sol
IEC	IERC20Detailed.sol	staking/contracts/interfaces/IERC20Detailed.sol
IFL	IFlashLoanReceiver.sol	lending/contracts/flashloan/interfaces/IFlashLoanReceiver.sol
ILR	ILendingRateOracle.sol	lending/contracts/interfaces/ILendingRateOracle.sol
IPG	IPriceOracleGetter.sol	lending/contracts/interfaces/IPriceOracleGetter.sol
IED	IERC20DetailedBytes.sol	lending/contracts/misc/IERC20DetailedBytes.sol
ICA	IChainlinkAggregator.sol	lending/contracts/interfaces/IChainlinkAggregator.sol

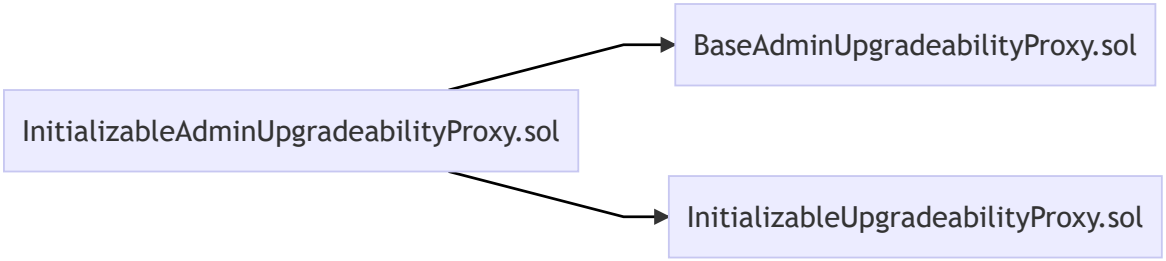
ID	Contract	Location
IMX	IbMXXDistributionManager.sol	staking/contracts/interfaces/IbMXXDistributionManager.sol
IMI	IbMXXIncentivesController.sol	staking/contracts/interfaces/IbMXXIncentivesController.sol
IKN	IKyberNetworkProxyInterface.sol	lending/contracts/interfaces/IKyberNetworkProxyInterface.sol
IRI	IReserveInterestRateStrategy.sol	lending/contracts/interfaces/IReserveInterestRateStrategy.sol
ILA	ILendingPoolAddressesProvider.sol	lending/contracts/interfaces/ILendingPoolAddressesProvider.sol
IUP	InitializableUpgradeabilityProxy.sol	staking/contracts/lib/InitializableUpgradeabilityProxy.sol
LEN	InitializableUpgradeabilityProxy.sol	lending/contracts/libraries/openzeppelin-upgradeability/InitializableUpgradeabilityProxy.sol
IAU	InitializableAdminUpgradeabilityProxy.sol	lending/contracts/libraries/openzeppelin-upgradeability/InitializableAdminUpgradeabilityProxy.sol
IAP	InitializableAdminUpgradeabilityProxy.sol	staking/contracts/lib/InitializableAdminUpgradeabilityProxy.sol
LPL	LendingPool.sol	lending/contracts/lendingpool/LendingPool.sol
LPC	LendingPoolCore.sol	lending/contracts/lendingpool/LendingPoolCore.sol
LEN	LendingPoolConfigurator.sol	lending/contracts/lendingpool/LendingPoolConfigurator.sol
LPD	LendingPoolDataProvider.sol	lending/contracts/lendingpool/LendingPoolDataProvider.sol
LPA	LendingPoolAddressesProvider.sol	lending/contracts/configuration/LendingPoolAddressesProvider.sol
LPM	LendingPoolLiquidationManager.sol	lending/contracts/lendingpool/LendingPoolLiquidationManager.sol
LPP	LendingPoolParametersProvider.sol	lending/contracts/configuration/LendingPoolParametersProvider.sol
MTN	MToken.sol	lending/contracts/tokenization/MToken.sol
MCR	MxxConverter.sol	converter/eth/contracts/MxxConverter.sol
ME0	MintableErc20.sol	staking/contracts/utils/MintableErc20.sol
CON	MxxConverterRedemption.sol	converter/bsc/contracts/MxxConverterRedemption.sol
PRO	Proxy.sol	lending/contracts/libraries/openzeppelin-upgradeability/Proxy.sol
PRX	Proxy.sol	staking/contracts/lib/Proxy.sol
RVT	RewardVault.sol	lending/contracts/vaults/RewardVault.sol
RMR	RewardsManager.sol	lending/contracts/rewards/RewardsManager.sol
SMH	SafeMath.sol	staking/contracts/lib/SafeMath.sol
SER	SafeERC20.sol	staking/contracts/lib/SafeERC20.sol
SMX	StakedbMXX.sol	staking/contracts/stake/StakedbMXX.sol
STN	StakedToken.sol	staking/contracts/stake/StakedToken.sol
USE	UintStorage.sol	lending/contracts/configuration/UintStorage.sol
UPY	UpgradeabilityProxy.sol	staking/contracts/lib/UpgradeabilityProxy.sol
UPX	UpgradeabilityProxy.sol	lending/contracts/libraries/openzeppelin-upgradeability/UpgradeabilityProxy.sol
VIE	VersionedInitializable.sol	lending/contracts/libraries/openzeppelin-upgradeability/VersionedInitializable.sol
VIL	VersionedInitializable.sol	staking/contracts/utils/VersionedInitializable.sol

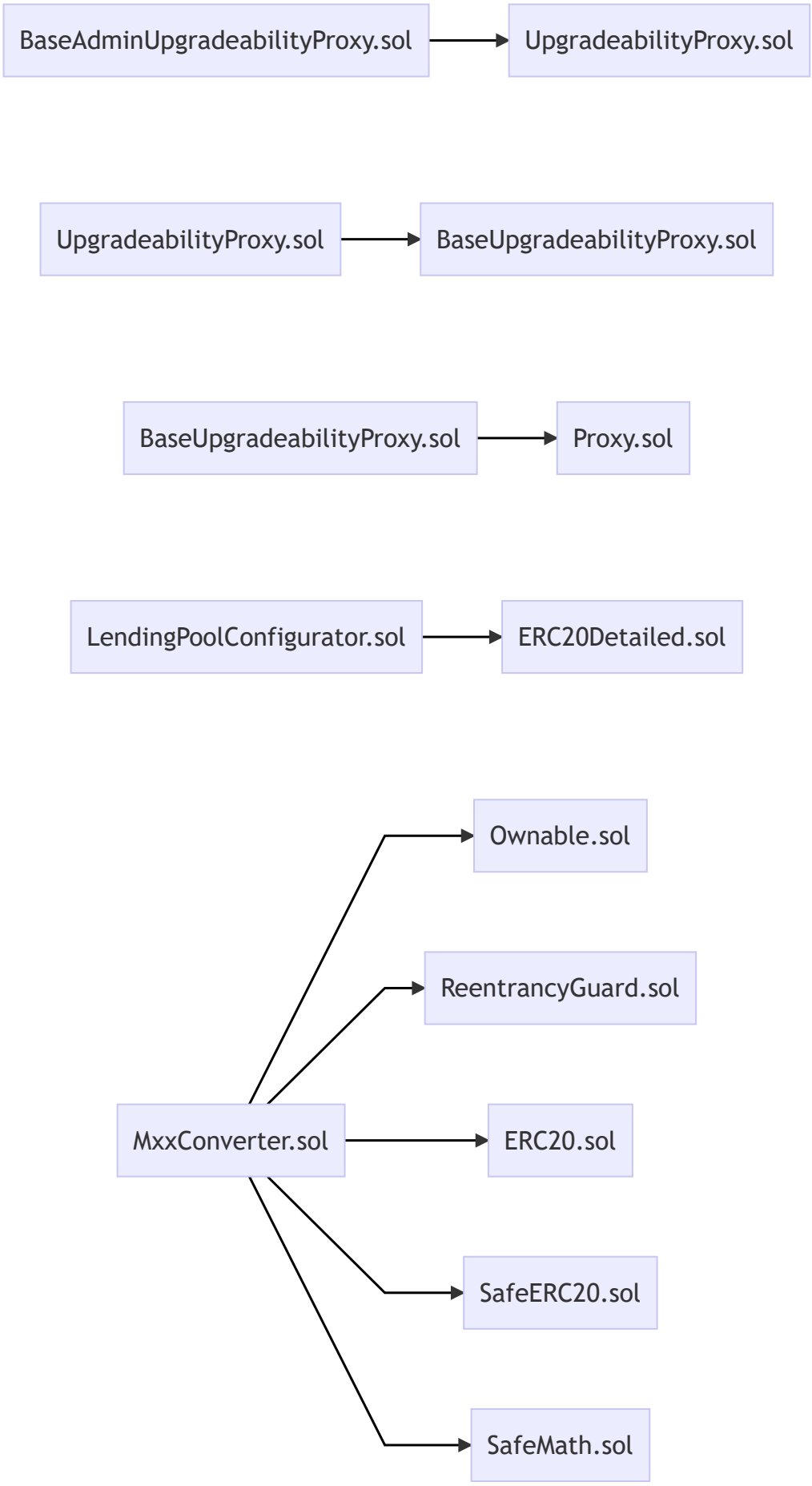
ID	Contract	Location
WRM	WadRayMath.sol	lending/contracts/libraries/WadRayMath.sol
WBP	WalletBalanceProvider.sol	lending/contracts/misc/WalletBalanceProvider.sol
BMX	bMxx.sol	converter/bsc/contracts/bMxx.sol
MXX	bMXXDistributionManager.sol	staking/contracts/stake/bMXXDistributionManager.sol
MXI	bMXXIncentivesController.sol	staking/contracts/stake/bMXXIncentivesController.sol

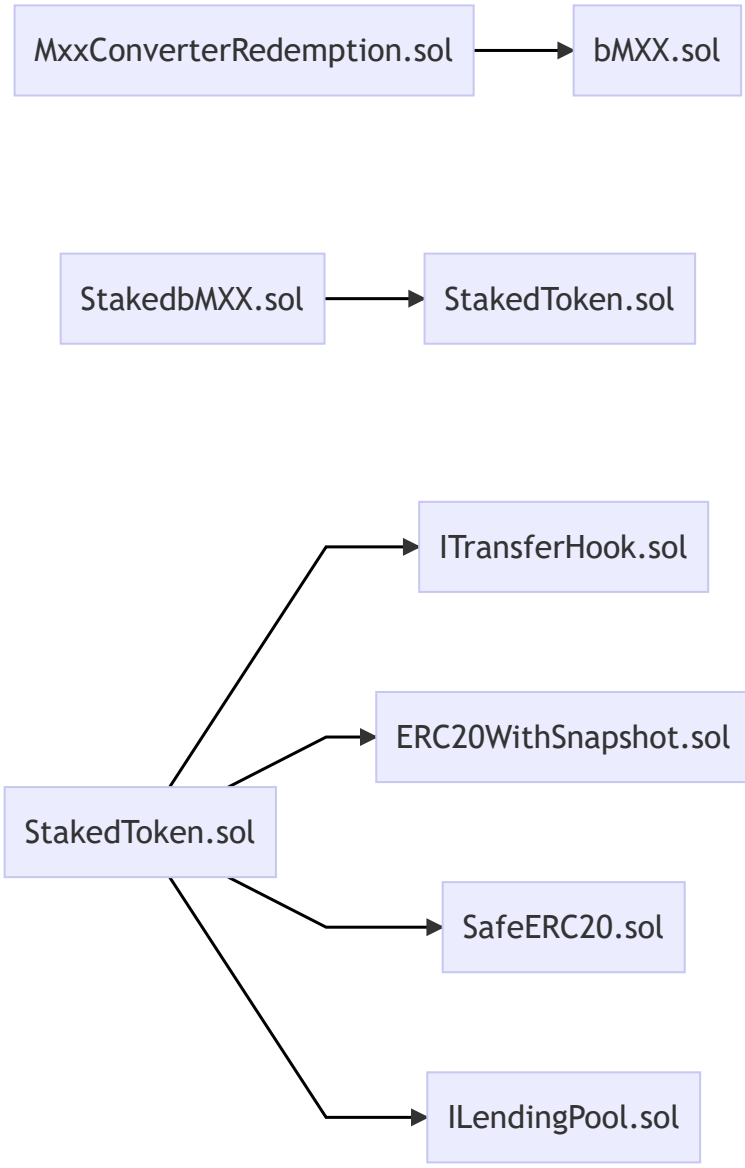
File Dependency Graph (BETA)

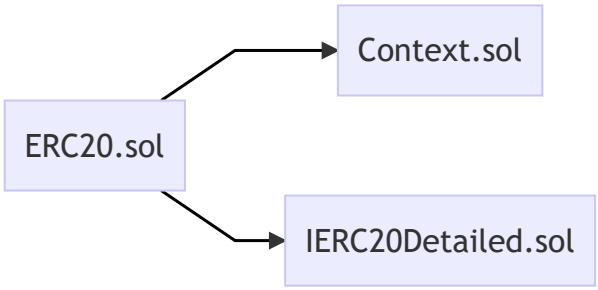


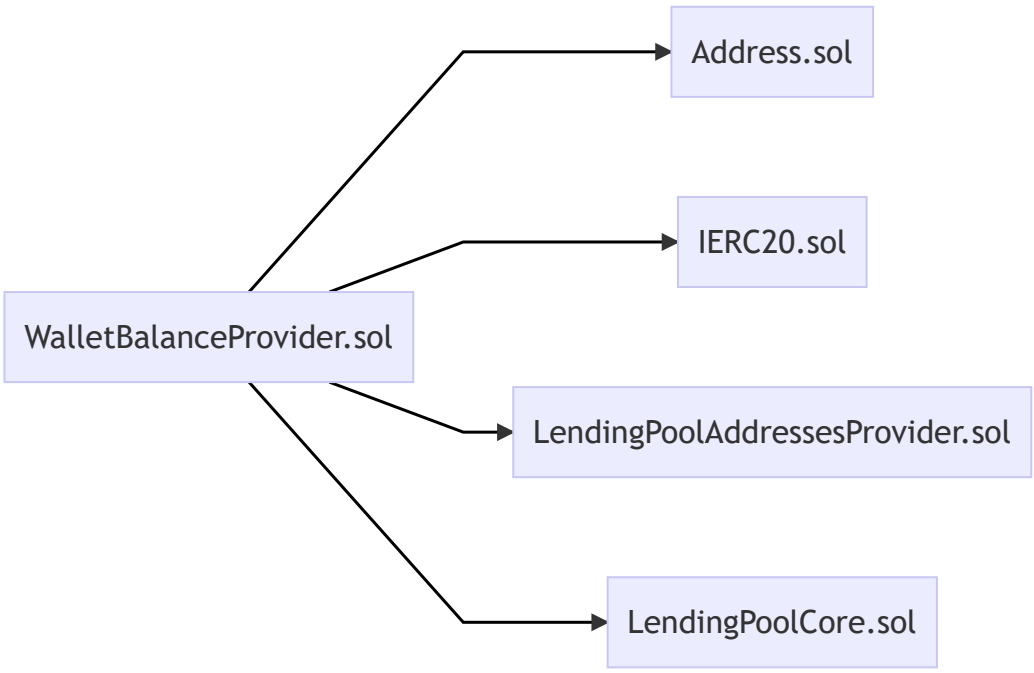


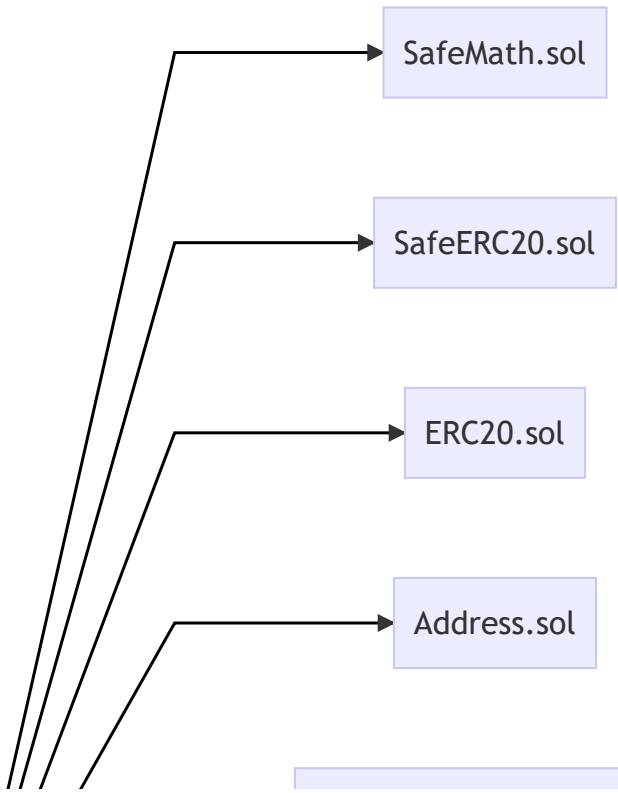


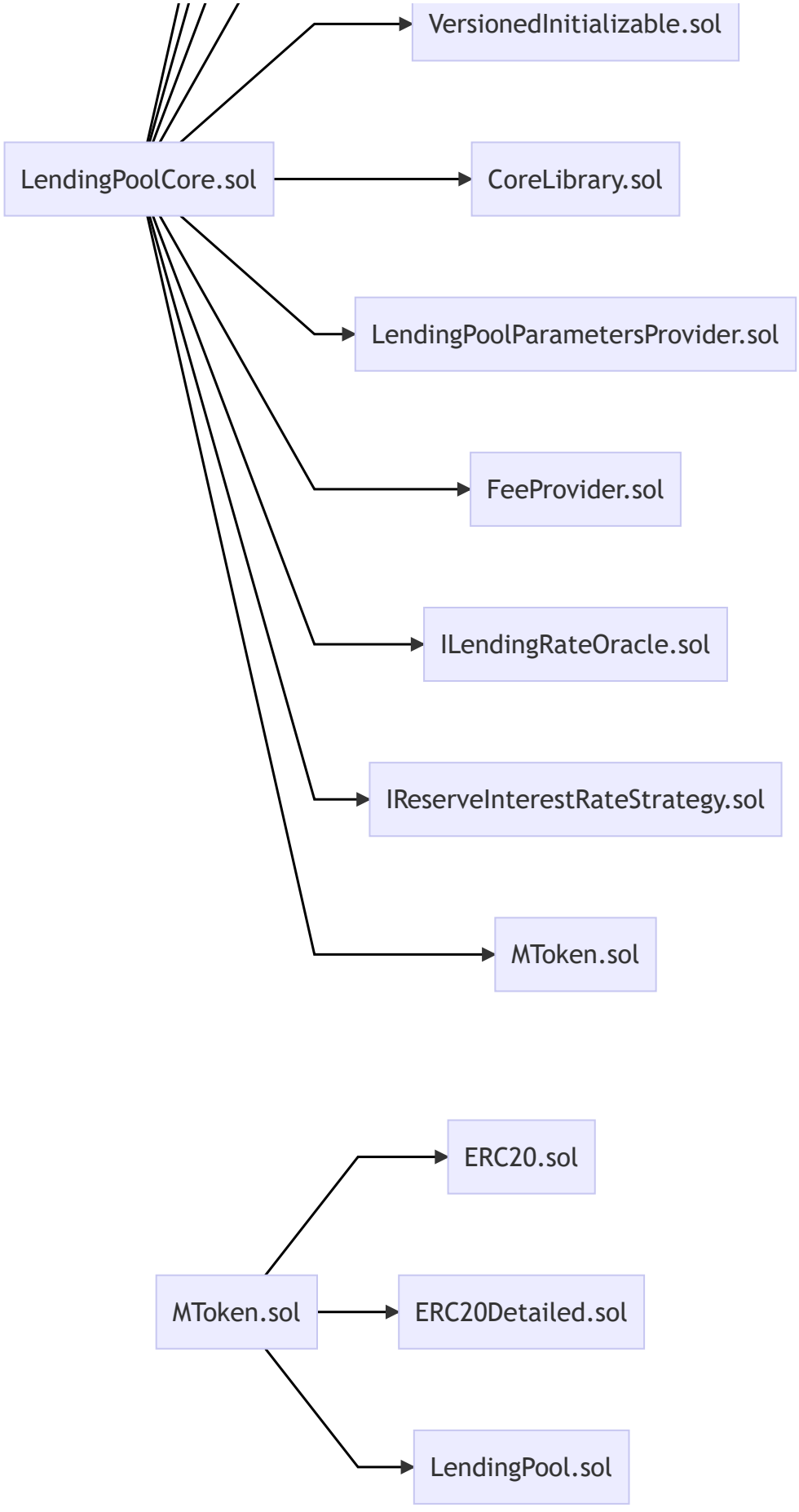

BaseAdminUpgradeabilityProxy.sol
UpgradeabilityProxy.sol
UpgradeabilityProxy.sol
BaseUpgradeabilityProxy.sol
BaseUpgradeabilityProxy.sol
Proxy.sol
LendingPoolConfigurator.sol
ERC20Detailed.sol
Ownable.sol
ReentrancyGuard.sol
MxxConverter.sol
ERC20.sol
SafeERC20.sol
SafeMath.sol

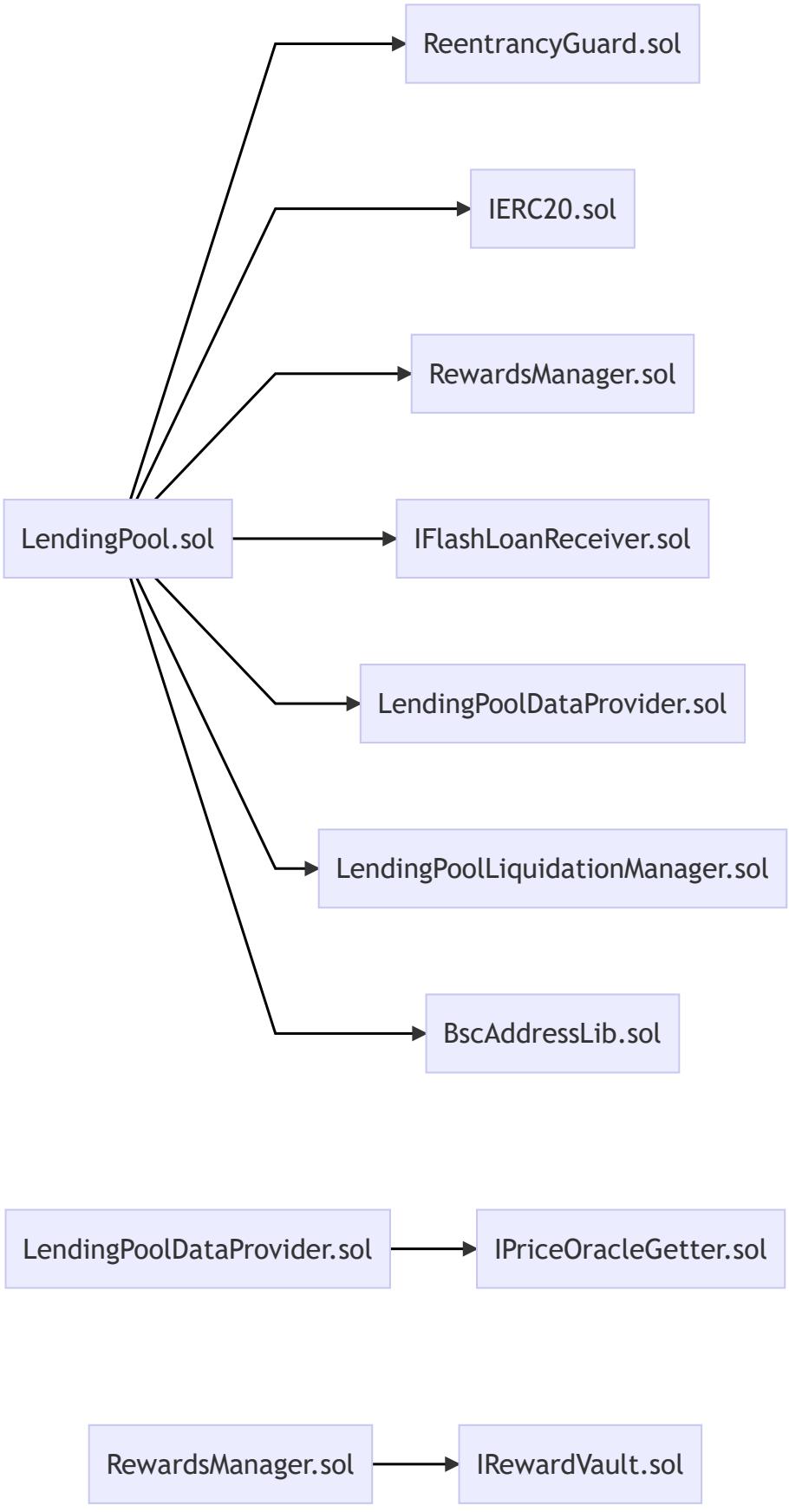

MxxConverterRedemption.sol
bMXX.sol
StakedbMXX.sol
StakedToken.sol
ITransferHook.sol
ERC20WithSnapshot.sol
StakedToken.sol
SafeERC20.sol
ILendingPool.sol
SafeERC20.sol
Address.sol
ERC20WithSnapshot.sol
ERC20.sol

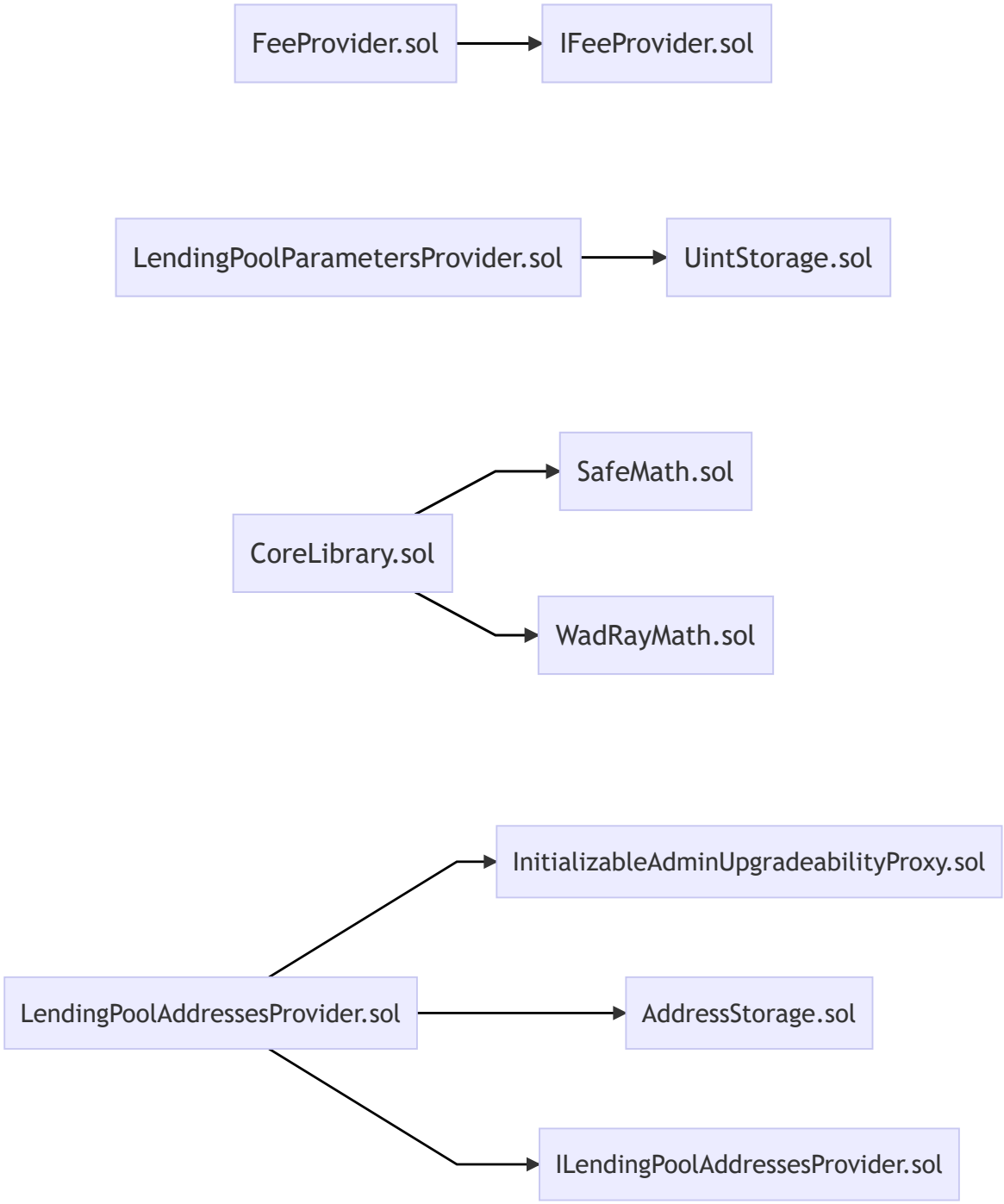

Context.sol
ERC20.sol
IERC20Detailed.sol

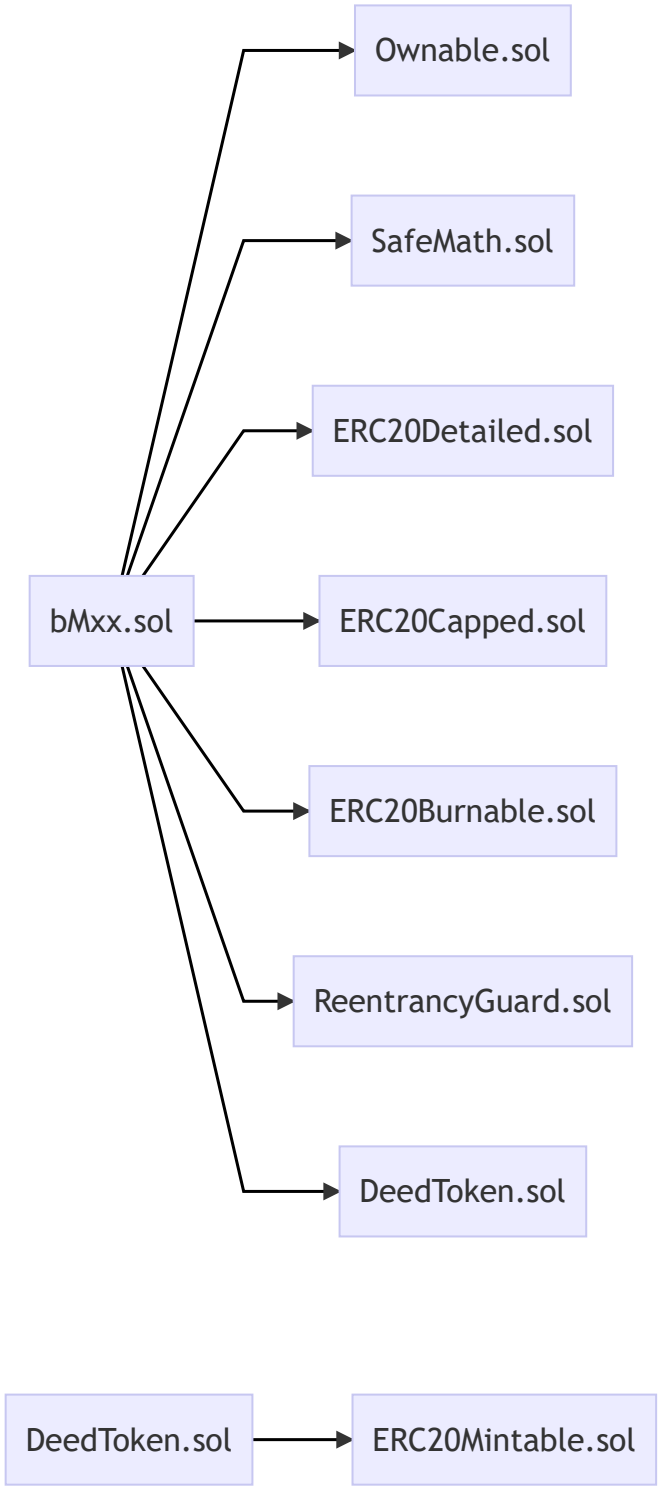

Address.sol
IERC20.sol
WalletBalanceProvider.sol
LendingPoolAddressesProvider.sol
LendingPoolCore.sol

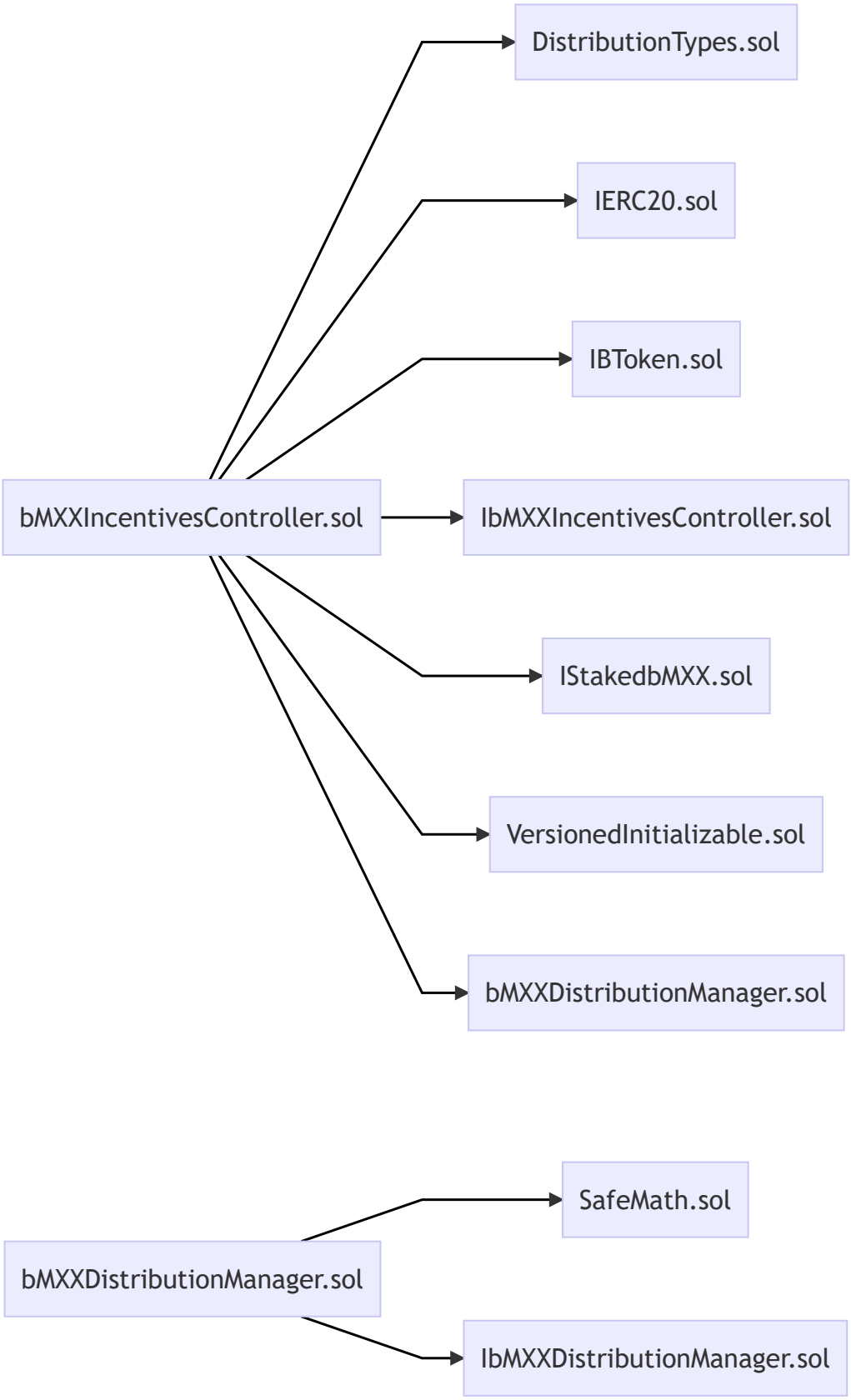

SafeMath.sol
SafeERC20.sol
ERC20.sol
Address.sol


VersionedInitializable.sol
LendingPoolCore.sol
CoreLibrary.sol
LendingPoolParametersProvider.sol
FeeProvider.sol
ILendingRateOracle.sol
IReserveInterestRateStrategy.sol
MToken.sol
ERC20.sol
MToken.sol
ERC20Detailed.sol
LendingPool.sol

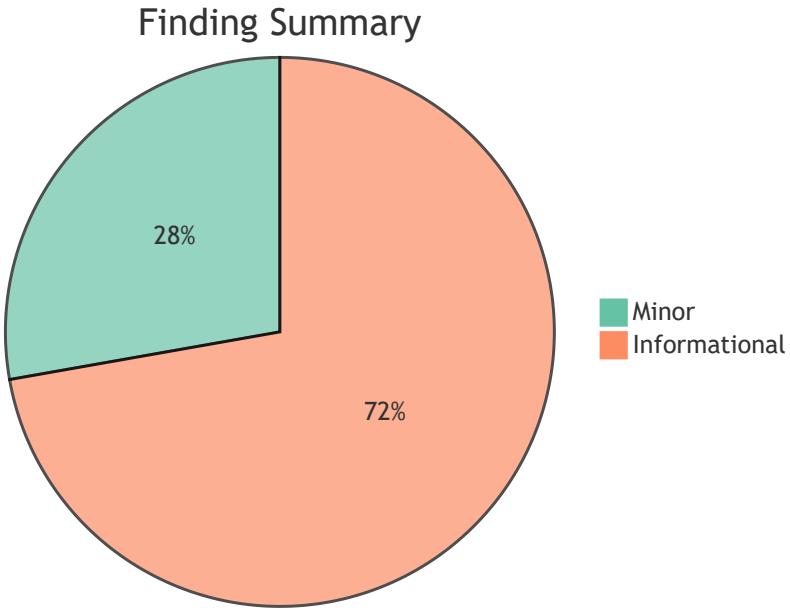

ReentrancyGuard.sol
IERC20.sol
RewardsManager.sol
LendingPool.sol
IFlashLoanReceiver.sol
LendingPoolDataProvider.sol
LendingPoolLiquidationManager.sol
BscAddressLib.sol
LendingPoolDataProvider.sol
IPriceOracleGetter.sol
RewardsManager.sol
IRewardVault.sol


FeeProvider.sol
IFeeProvider.sol
LendingPoolParametersProvider.sol
UintStorage.sol
SafeMath.sol
CoreLibrary.sol
WadRayMath.sol
InitializableAdminUpgradeabilityProxy.sol
LendingPoolAddressesProvider.sol
AddressStorage.sol
ILendingPoolAddressesProvider.sol


Ownable.sol
SafeMath.sol
ERC20Detailed.sol
bMxx.sol
ERC20Capped.sol
ERC20Burnable.sol
ReentrancyGuard.sol
DeedToken.sol
DeedToken.sol
ERC20Mintable.sol


DistributionTypes.sol
IERC20.sol
IBToken.sol
bMXXIncentivesController.sol
IbMXXIncentivesController.sol
IStakedbMXX.sol
VersionedInitializable.sol
bMXXDistributionManager.sol
SafeMath.sol
bMXXDistributionManager.sol
IbMXXDistributionManager.sol

Findings



ID	Title	Type	Severity	Resolved
RMR-01	Enum values are not used	Volatile Code	Minor	✓
RMR-02	Outdated Dependency	Language Specific	Minor	✓
RMR-03	Single Member Structs	Gas Optimization	Informational	✓
RMR-04	Unaccounted Truncated Reward	Mathematical Operations	Minor	✓
LPP-01	Mutability Specifiers Missing	Gas Optimization	Informational	✓
FPR-01	Variable Mutability Specifiers	Gas Optimization	Informational	✓
FPR-02	Redundant `SafeMath` Utilization	Gas Optimization	Informational	✓
LPL-01	Visibility Specifiers Missing	Language Specific	Informational	✓
LPL-02	Inexplicable Deviation from Original Codebase	Logical Issue	Minor	✓
LPL-03	Uncommon Naming Conventions	Coding Style	Informational	✓
LPL-04	Unused Struct Member	Gas Optimization	Informational	✓
LPL-05	Redundant Self-External Call	Gas Optimization	Informational	✓
LPC-01	Conditional Optimization	Gas Optimization	Informational	✓
LPM-01	Visibility Specifiers Missing	Language Specific	Informational	✓
CLY-01	Incorrect Comment	Coding Style	Informational	✓
RVT-01	Visibility Specifiers Missing	Language Specific	Informational	✓
RVT-02	Unguarded Fallback Function	Control Flow	Minor	✓

ID	Title	Type	Severity	Resolved
RVT-03	Centralized Control of Funds	Logical Issue	Informational	✓

RMR-01: Enum values are not used

Type	Severity	Location
Volatile Code	Minor	RewardsManager.sol L152, L160

Description:

Using 0 and 1 directly could lead to problems as an enum values order change will break this piece of code.

Recommendation:

We would recommend using the result of uint casting from the enum directly

Alleviation:

The numeric literals were properly replaced by their `enum` representation.

RMR-02: Outdated Dependency

Type	Severity	Location
Language Specific	Minor	RewardsManager.sol L3-L6

Description:

The linked `import` statement uses a direct `sol` file dependency by specifying the GitHub link used to access it.

Recommendation:

The link provided points to `v2.4.0` of the OpenZeppelin contract repository which is an outdated and no longer maintained codebase. We advise that the `import` statement is instead adjusted to utilize a local dependency management system, such as `npm`, and to utilize one of the latest and stable versions of OpenZeppelin's libraries. We have observed the same issue across the codebase in numerous contracts and as such, we advise the same fix is applied to every one as it will not be listed independently.

Alleviation:

The team stated that upgrading the dependency breaks the functionality of the contracts and deviates from the dependencies utilized by the original project, AAVE. Thus, the team will proceed with not updating the dependencies.

RMR-03: Single Member Structs

Type	Severity	Location
Gas Optimization	Informational	RewardsManager.sol L37-L39, L56-L58

Description:

The linked structs contain a single member which points to a statically sized struct array.

Recommendation:

In this scenario, using a struct declaration is redundant and a direct declaration can be used instead.

Alleviation:

The codebase was refactored so as to render the single member struct redundant.

RMR-04: Unaccounted Truncated Reward

Type	Severity	Location
Mathematical Operations	Minor	RewardsManager.sol L343

Description:

The reward mechanism uses a proportionate reward calculation that can lead to imperfect portions and thus dust being left unclaimed due to the strict conditional that ensures the reward left is sufficient to cover the current claim.

Recommendation:

Instead of moving on to the next reward via `continue`, additional checks should be imposed that ensure both amounts are non-zero and then the `if` clause within should simply assign `rewardAmt` to `r.amount.sub(r.paidSoFar)` to ensure the full amount is claimable.

Alleviation:

The `if` clause correctly assigns the `amountAvailable` to `rewardAmt` if it exceeds the said value due to mathematical truncations, similarly to how other DeFi projects operate.

LPP-01: Mutability Specifiers Missing

Type	Severity	Location
Gas Optimization	Informational	LendingPoolParametersProvider.sol L14-L15

Description:

The linked variables are assigned to only once, either during their contract-level declaration or during the `constructor`'s execution.

Recommendation:

For the former, we advise that the `constant` keyword is introduced in the variable declaration to greatly optimize the gas cost involved in utilizing the variable. For the latter, we advise that the `immutable` mutability specifier is set at the variable's contract-level declaration to greatly optimize the gas cost of utilizing the variables. Please note that the `immutable` keyword only works in Solidity versions `v0.6.5` and up.

Alleviation:

The variables were properly set to be `constant`.

FPR-01: Variable Mutability Specifiers

Type	Severity	Location
Gas Optimization	Informational	FeeProvider.sol L17, L18, L21, L22, L39-L42

Description:

Currently, the linked variables are assigned to value literals during the execution of the `initialize` function of the contract.

Recommendation:

As these values are literals, their assignment can instead be relocated to the declarations themselves, allowing them to be set as `constant` thus optimizing the overall gas cost of the contract.

Alleviation:

The variables were set to not be assigned during `initialize` and instead be assigned at their declarations, allowing them to be set to `constant`.

FPR-02: Redundant `SafeMath` Utilization

Type	Severity	Location
Gas Optimization	Informational	FeeProvider.sol L83

Description:

The linked `SafeMath` invocation conducts two consequent `sub` invocations on the `safetyModuleRate` which is meant to represent 100% of a particular ratio whereas the values subtracted are meant to represent a sub-100% ratio that fundamentally cannot exceed in sum the 100% of the full amount.

Recommendation:

As the values of `supplierRewardRate` and `governanceRewardRate` will never exceed, in sum, the value of `safetyModuleRate` the `SafeMath` utilization here can safely be omitted.

Alleviation:

The redundant usage of `SafeMath` was omitted from the codebase.

LPL-01: Visibility Specifiers Missing

Type	Severity	Location
Language Specific	Informational	LendingPool.sol L38

Description:

The linked variable declarations do not have a visibility specifier explicitly set.

Recommendation:

Inconsistencies in the default visibility the Solidity compilers impose can cause issues in the functionality of the codebase. We advise that visibility specifiers for the linked variables are explicitly set.

Alleviation:

The `public` visibility specifier was added to the variable declaration.

LPL-02: Inexplicable Deviation from Original Codebase

Type	Severity	Location
Logical Issue	Minor	LendingPool.sol L961-L969

Description:

The original codebase does not check the `success` variable and immediately `revert`s, signifying that a "successful" execution can be achieved with a "failed" `returnCode` which should throw. The new code adjusts the error handling to allow valid execution of `liquidationCall` even if the `returnCode` is non-zero.

Recommendation:

We advise that the original error handling method is set, or, if the new handling method is desired that its proper function is ensured and that the state changes within `liquidationCall` of `liquidationManager` are impermanent.

Alleviation:

The team adjusted the codebase to properly `revert` in case the `returnCode` is non-zero.

LPL-03: Uncommon Naming Conventions

Type	Severity	Location
Coding Style	Informational	LendingPool.sol L231, L404, L585, L1090, L1268

Description:

The linked variable declarations either contain typos or do not conform to the original project's style guide with regards to their naming convention. Additionally, the declaration of L1090 should not be set to the `bMXX` address as it is meant to represent the `mTokenAddress`.

Recommendation:

We advise that the issues pointed out in the description are remediated.

Alleviation:

The issue was partially alleviated by addressing a portion of the specified issues.

LPL-04: Unused Struct Member

Type	Severity	Location
Gas Optimization	Informational	LendingPool.sol L596

Description:

The `RewardLocalVars` struct contains an unused `struct` member, specifically the `safety` aspect, which reserves a full 32-byte slot in memory redundantly.

Recommendation:

We advise that it is safely omitted from the `struct` declaration as the `safety` portion can be calculated by subtracting the other two portions from the whole percentage.

Alleviation:

The struct member was omitted from the struct's declaration.

LPL-05: Redundant Self-External Call

Type	Severity	Location
Gas Optimization	Informational	LendingPool.sol L655

Description:

When a function call to self is preceded by the `this` keyword, it is treated as an external call to self replacing the ephemeral state variables such as the `msg.sender` member.

Recommendation:

The `updateRewards` function invoked here does not utilize any state variables as it solely contains external calls, rendering the self-external call redundant.

Alleviation:

The external call specifier (`this`) was omitted from the linked function invocation.

LPC-01: Conditional Optimization

Type	Severity	Location
Gas Optimization	Informational	LendingPoolCore.sol L536-L573

Description:

The inner conditional of `_transferFromCore` can be moved upwards to only be conducted once in all three transfers as it does not change between the conditional checks of L537, L549 and L561.

Recommendation:

We advise that the conditionals are refactored as detailed in the description to optimize the gas cost of the function.

Alleviation:

The conditionals were optimized according to the nesting structure specified in the exhibit.

LPM-01: Visibility Specifiers Missing

Type	Severity	Location
Language Specific	Informational	LendingPoolLiquidationManager.sol L31-L33

Description:

The linked variable declarations do not have a visibility specifier explicitly set.

Recommendation:

Inconsistencies in the default visibility the Solidity compilers impose can cause issues in the functionality of the codebase. We advise that visibility specifiers for the linked variables are explicitly set.

Alleviation:

The `public` visibility specifiers were set to the linked contract-level declarations properly.

CLY-01: Incorrect Comment

Type	Severity	Location
Coding Style	Informational	CoreLibrary.sol L68

Description:

The comment specifies that the `address` stored in the declaration below is the `bMXXToken` address, whereas it is the `MToken` address.

Recommendation:

The comment should be adjusted accordingly.

Alleviation:

The comment was properly adjusted to reflect what is represented by the struct's member.

RVT-01: Visibility Specifiers Missing

Type	Severity	Location
Language Specific	Informational	RewardVault.sol L13

Description:

The linked variable declarations do not have a visibility specifier explicitly set.

Recommendation:

Inconsistencies in the default visibility the Solidity compilers impose can cause issues in the functionality of the codebase. We advise that visibility specifiers for the linked variables are explicitly set.

Alleviation:

A visibility specifier was properly introduced for the `paused` variable.

RVT-02: Unguarded Fallback Function

Type	Severity	Location
Control Flow	Minor	RewardVault.sol L21

Description:

The current fallback implementation allows accidental ETH transfers to the contract by unsuspecting EOA (Externally Owned Accounts).

Recommendation:

We advise that a `require` check is imposed that either simply checks the sender of the funds is a contract or even better checks that it is the intended sender the contract is solely meant to receive transfers from.

Alleviation:

A `require` check was added ensuring that accidental transfers are rejected by the contract.

RVT-03: Centralized Control of Funds

Type	Severity	Location
Logical Issue	Informational	RewardVault.sol L98-L104

Description:

While the purpose of this emergency function is to act as a failsafe in case of failure of the contract and allow retrieval of funds, it is immediately enforceable and could be used maliciously.

Recommendation:

We advise that a cooldown period is set instead, as is the standard followed by the space, whereby an emergency measure is announced, users are allowed to still attempt to withdraw their funds / rewards in a specified period i.e. a week and then the emergency function allows the owner to withdraw the funds. This accommodates for all use cases and cannot be abused by any party.

Alleviation:

An emergency cooldown system was introduced whereby emergency withdrawals are offset by a cooldown thus allowing users to properly intract with the contract before administrative action is taken on withdrawals.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that 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.

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

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

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

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

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a `struct` assignment operation affecting an in-memory `struct` rather than an in-storage one.

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 and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a `constructor` assignment imposing different `require` statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as `constant` contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Dead Code

Code that otherwise does not affect the functionality of the codebase and can be safely omitted.