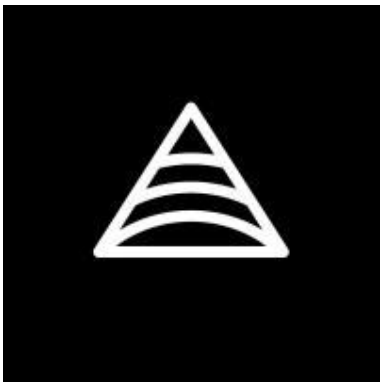

Aries

Smart Contract

Audit Report

MOVEBIT

contact@movebit.xyz

https://twitter.com/movebit_

06/14/2023

Aries Smart Contract Audit Report

1 Executive Summary

1.1 Project Information

Description	A decentralized lending protocol that supports deposit and withdrawal services, lending and borrowing services, flash loans services, and unhealthy profile liquidation services.
Type	Lending
Auditors	MoveBit
Timeline	Apr 19, 2023 – May 2, 2023; June 8, 2023 – June 14, 2023
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/Aries-Markets/protocol-sui
Commits	1870f804bbce90fc24f29f9e0409c65e2240ab29 f25e147bc3c7303e3cfd38e305ac6db79c8b14ef 69207e3f9383d8b0be80d06b70d4f3010a590cfa d0fd66e8b0f33e2df78b5d9ca5b3d1ecbda0dde5

1.2 Files in Scope

The following are the SHA1 hashes of the last reviewed files.

ID	Files	SHA-1 Hash
CTR	packages/aries-new/sources/controller.move	4c8055eaa8a1259acd6552788b0752a6cc630cd7
ORC	packages/aries-new/sources/oracle.move	457dfd0721ba7c1c9bbc74e0bea6c66db68802bf
DEC	packages/aries-new/sources/decimal/decimal.move	a76b5d009e96c75986d950d85c45dae2b4d88059
UMT	packages/aries-new/sources/decimal/u128_math.move	bc82ae7cc7331bfab73b01d0fab430d4f8d6c678
LEND	packages/aries-new/sources/lending/lending.move	221c674c9209aecce6de99b0876759f51cfdc977
LQD	packages/aries-new/sources/lending/liquidate.move	cfd147ac4ae3f132bde61b262981dc6597b7ad37
MKT	packages/aries-new/sources/lending/market.move	9d85b4e0e5853d8dce12da30a96575099ed6be7c
PFT	packages/aries-new/sources/lending/profile_status.move	c3fc798ba299e1a6166e519cbff80f009ddf805a
COMPI	packages/aries-new/sources/lending/interest/compound_interest.move	f524b33db0bf49b3b0d438f144c848cd76069b77
ITR	packages/aries-new/sources/lending/interest/interest.move	e137e00ccf621e250890bdfb40fe00399bf8d06e
PRF	packages/aries-new/sources/lending/profile/profile.move	6fadb7d34c9e5860715d73ef803825700670df09
RSVCFG	packages/aries-new/sources/lending/reserve/reserve_config.move	749d4d853036d7887bca1967d6eede4c4cdd5e31

RSVST	packages/aries-new/sources/lending/reserve/reserve_status.move	04b5dd23a572d514f7ec97371d77cea8af3e8f7c
RSV	packages/aries-new/sources/lending/reserve/reserve.move	ad78932cdfe5ab7d6295a76369256169c58500b5
FKB	packages/aries-new/sources/utils/fakebtc.move	43bdf0370e3954765148c33c7f95db078f6c3e34
FKU	packages/aries-new/sources/utils/fakeusdc.move	760f59f9fe87c755cce47f0705ae07225e649490
MTU	packages/aries-new/sources/utils/math_util.move	d2979f2de97d32c4a6ff2fa1c19d6458073803cc
XMT	packages/aries-new/Move.toml	68953684d6c8029965d47b2a17a29d8a66d6f212

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	24	20	4
Informational	2		2
Minor	11	11	
Medium	9	7	2
Major	2	2	
Critical			

1.4 MoveBit Audit BreakDown

MoveBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not

limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow by bit operations
- Number of rounding errors
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting
- Unchecked CALL Return Values
- The flow of capability
- Witness Type

1.5 Methodology

The security team adopted the "**Testing and Automated Analysis**", "**Code Review**" and "**Formal Verification**" strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope sees in section **1.2**.

(3) Formal Verification

Perform formal verification for key functions with the Move Prover.

(4) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by **Aries** to identify any potential issues and vulnerabilities in the source code of the **Aries** smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we have identified **24** issues of varying severity, listed below.

ID	Title	Severity	Status
GLOBAL-01	Third-Party Dependency	Medium	Fixed
RSVCFG-01	Missing Key Validation in `ReserveConfig`	Medium	Fixed
RSVCFG-02	Lack of Input Validation	Medium	Fixed
RSVCFG-03	Misspelled Keywords	Minor	Fixed
RSVCFG-04	Function Logic Does Not Match the Annotation	Minor	Fixed
CTR-01	Unreasonable Repayment Logic for Flash Loans	Medium	Fixed
CTR-02	Unchecked Return	Minor	Fixed
CTR-03	Inconsistent Error Code Format in Assert Statements	Minor	Fixed
MKT-01	Centralization Risk	Medium	Acknowledged
MKT-02	Incorrect Function Call Permissions	Minor	Fixed

MKT-03	The Addition of `reserve` is Missing Validation	Medium	Acknowledged
MKT-04	Lack of Market Version Check	Major	Fixed
MKT-05	Validation is Required Before Deleting `reserve_addr`	Medium	Fixed
MKT-06	Optimization of Function Visibility	Minor	Fixed
LEND-01	Premature Assertion Checks	Minor	Fixed
LEND-02	Unverified Amounts Being Set to 0	Minor	Fixed
LEND-03	Reserve Interest Not Updated in a Timely Manner	Medium	Fixed
ORC-01	Unused or Improperly Used Functions and Constants	Minor	Fixed
ORC-02	Missing Validation While Updating Oracle Price	Medium	Fixed
DEC-01	Overflow Risk in Utility Functions	Minor	Fixed
RSVST-01	Incorrect Event Timestamp Parameter Setting	Minor	Fixed
PRF-01	Variable Naming is Not Standardized	Informational	Acknowledged
PRF-02	The Variable Name Does Not Match its Function	Informational	Acknowledged
LQD-01	Missing Market Checks	Major	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the `Aries Sui Lending` Smart Contract:

Admin

- Admin can create a `market` through `add_market()`.

- Admin can add a `reserve` for a certain Coin through `add_reserve()`.
- Admin can set the `compound_interest` configuration through `set_compound_interest_cfg()`.
- Admin can update `reserve` configuration through `update_reserve_config()`.

Users

- Users can add a `profile` for themselves through `add_profile()`.
- Users can deposit tokens into the `market` through `deposit()`.
- Users can withdraw the deposited tokens from the `market` through `withdraw()`.
- Users can borrow tokens from the `market` through `borrow()`.
- Users can repay the borrowed tokens into the `market` through `repay()`.
- Users can liquidate the unhealthy `profile` through `liquidate()`.
- Users can begin a flash_loan through `begin_flash_loan()`.
- Users can end a flash_loan through `end_flash_loan()`.
- Users can withdraw the deposited tokens from the `market` and borrow tokens from the `market` through `withdraw_with_borrow()`.
- Users can repay the borrowed tokens into the `market` and deposit tokens into the `market` through `deposit_with_repay()` in the meanwhile.

4 Findings

GLOBAL-01 Third-Party Dependency

Severity: Medium

Status: Fixed

Descriptions: During the audit process, we discovered that the system relies on third-party services for certain functionalities, such as PriceOracle. However, please note that this audit does not cover third-party dependencies, including PriceOracle. We assume that the data provided by the oracle is accurate and properly handled by the system.

Suggestion: It is recommended to utilize audited and widely adopted third-party dependencies whenever possible. Necessary security measures should be implemented to address potential issues that may arise from these dependencies. Additionally, proactive monitoring of the third-party services is essential during the operational phase to promptly detect and mitigate any potential risks and avoid potential losses.

Resolution: The customer informed us that the third-party dependency has already been audited and can be guaranteed to be free of issues.

RSVCFG-01 Missing Key Validation in `ReserveConfig`

Severity: Medium

Status: Fixed

Code Location: packages/aries-new/sources/lending/reserve/reserve_config.move#L85.
Descriptions: In the `reserve_config::build_config()` function, the `ReserveConfig` update does not check if the `liquidation_threshold` value is higher than the `loan_to_value`. When an incorrect configuration is entered, in which the `liquidation_threshold` value is lower than the `loan_to_value`, it can cause the asset to be immediately liquidated when the borrowing amount is close to the borrowing capacity, resulting in the loss of user assets.

Suggestion: Add the assertion `assert!(loan_to_value < liquidation_threshold, ERESERVE_CONFIG_VIOLATION);` when configuring the ReserveConfig to prevent such errors.

Resolution: The client has followed our suggestion and fixed the issue.

RSVCFG-02 Lack of Input Validation

Severity: Medium

Status: Fixed

Code Location: packages/aries-new/sources/lending/reserve/reserve_config.move#L85.
Descriptions: The function `reserve_config::build_config()` does not validate the input parameter `reserve_ratio` to ensure that it is not greater than 100 when updating the `reserve_config`. This can result in incorrect calculations of the reserve value.

Suggestion: Add input validation to ensure that `reserve_ratio` is not greater than 100. This can be achieved by adding an `assert!(reserve_ratio <= 100, ERESERVE_CONFIG_VIOLATION);` statement within the `reserve_config::build_config()` function to check if the input is valid.

Resolution: The client has followed our suggestion and fixed the issue.

RSVCFG-03 Misspelled Keywords

Severity: Minor

Status: Fixed

Code Location: packages/aries-new/sources/lending/reserve/reserve_config.move#L69.
Descriptions: A misspelling error has been found in the code at line 69 of `reserve_config.move` where the keyword `#[test_only]` is misspelled as `#[test_onlu]`. This typing error can cause a failure in testing when executed.

Suggestion: Change `#[test_onlu]` to `#[test_only]`.

Resolution: The client has followed our suggestion and fixed the issue.

RSVCFG-04 Function Logic Does Not Match the Annotation

Severity: Minor

Status: Fixed

Code Location: packages/aries-new/sources/lending/reserve/reserve_config.move#L42, L46.
Descriptions: When `allow_collateral` and `allow_redeem` are set to false, the value of the user's collateral is 0, which will prevent the user from withdrawing, borrowing, etc., and the comment means whether to allow the use of collateral to continue borrowing, and there is a conflict between the two.

Suggestion: Modify the annotation or function to ensure that the function matches the annotation.

Resolution: The client has followed our suggestion and fixed the issue.

CTR-01 Unreasonable Repayment Logic for Flash Loans

Severity: Medium

Status: Fixed

Code Location: packages/aries-new/sources/controller.move#L209.
Descriptions: In the `end_flash_loan()` function within `controller.move`, when repayment is made for a flash loan, if the amount in `coin_src` exceeds the outstanding payment amount of the flash loan, the excess amount is used to repay other debts or make deposits. This is not a reasonable logic for flash loans.

Suggestion: When repaying a flash loan, only the amount borrowed in the current flash loan should be repaid, and previous deposits and loans should not be affected. We suggest calling the function `lending::repay` and passing in the amount borrowed for repayment.

Resolution: The client has followed our suggestion and fixed the issue.

CTR-02 Unchecked Return

Severity: Minor
Status: Fixed

Code Location: packages/aries-new/sources/controller.move#L13, L22, L82, L219;

packages/aries-new/sources/lending/reserve/reserve.move#L61, L87, L234, L355, L468;

packages/aries-new/sources/lending/lending.move#L75, L443, L373;

packages/aries-new/sources/lending/profile/profile.move#L230, L270;

packages/aries-new/sources/lending/interest/interest.move#L98, L102;

packages/aries-new/sources/utils/fakeusdc.move#L62;

packages/aries-new/sources/lending/liquidate.move#L176;

packages/aries-new/sources/lending/market.move#L301.

Descriptions: Certain library functions such as `coin::burn()`, `vec_map::remove()`, `table::remove()`, `bag::remove()`, `balance::join()`, and `balance::decrease_supply()` have return values, but are not being properly utilized in the code. Similarly, functions such as `market::init_market()`, `profile::add_profile()`, `market::add_reserve()`, `lending::deposit_with_repay()`, `reserve::repay()`, and `reserve::deposit_lp_coin()` also have return values but are not being properly utilized when called. This could result in important return values not being checked, potentially leading to security vulnerabilities.

Suggestion: Code developers should utilize these functions properly by checking their return values to ensure that no important information is being overlooked.

Resolution: The client has followed our suggestion and fixed the issue.

CTR-03 Inconsistent Error Code Format in Assert Statements

Severity: Minor

Status: Fixed

Code Location: packages/aries-new/sources/controller.move,#L220, L221;

packages/aries-new/sources/lending/profile/profile.move#L257;

packages/aries-new/sources/lending/reserve/reserve_config.move#L189.

Descriptions: The use of the integer value 0 as an error code in assert statements at line 257 of the file `profile.move`, line 189 of `reserve_config.move`, and lines 220-221 of `controller.move` violates the standard error code conventions. This inconsistency makes it difficult to convey accurate error information to users or developers.

Suggestion: Standard error code constants should be used instead of direct integer values.

Resolution: The client has followed our suggestion and fixed the issue.

MKT-01 Centralization Risk

Severity: Medium

Status: Acknowledged

Code Location: packages/aries-new/sources/lending/market.move.

Descriptions: All resource storage and modification is currently under the control of a single project administrator account, and this account is incapable of modification. This creates a significant centralization risk which could have negative consequences.

Suggestion: Implement an interface that allows for multiple administrators or a multi-signature account to manage resources to mitigate.

Resolution: The contract will be deployed with a multi-sig account.

MKT-02 Incorrect Function Call Permissions

Severity: Minor

Status: Fixed

Code Location: packages/aries-new/sources/lending/market.move#L96.

Descriptions: The function `market::new()` is only called in `market.move`, yet its visibility is public (friend).

Suggestion: Modify this function to a private function to ensure that it is not accessed improperly.

Resolution: The client has followed our suggestion and fixed the issue.

MKT-03 The Addition of `reserve` is Missing Validation

Severity: Medium

Status: Acknowledged

Code Location: packages/aries-new/sources/lending/market.move#L176.

Descriptions: The function `add_reserve()` is missing validation, which allows the addition of reserves with the same coin.

Suggestion: Please add an assert validation to prevent the addition of reserves with the same coin in the function `add_reserve()`.

Resolution: The client believes that this is not a problem as it is controlled by the administrator.

MKT-04 Lack of Market Version Check

Severity: Major

Status: Fixed

Code Location: packages/aries-new/sources/lending/market.move#L273.

Descriptions: The function `set_reserve_config_obj()` is missing validation for the market version, which may result in inconsistent market information.

Suggestion: Please add `latest_market_or_throw(market);` to validate the market version in the function `set_reserve_config_obj()`.

Resolution: The client has followed our suggestion and fixed the issue.

MKT-05 Validation is Required Before Deleting `reserve_addr`

Severity: Medium

Status: Fixed

Code Location: packages/aries-new/sources/lending/market.move#L321.

Descriptions: The function `set_reserve_config()` lacks validation for whether `reserve_addr` already exists before deletion, which may cause transaction failure and prevent the setting up of a new `reserve_config`.

Suggestion: Please validate whether `reserve_addr` exists before its deletion in the function `set_reserve_config()`. If it does not exist, then do not proceed with the deletion logic.

Resolution: The client has followed our suggestion and fixed the issue.

MKT-06 Optimization of Function Visibility

Severity: Minor

Status: Fixed

Code Location: packages/aries-new/sources/lending/market.move#L410, L420.

Descriptions: The functions `get_reserve_detail_mut_by_address()` and `get_reserve_detail_mut()` return mutable objects. It is best not to set this type of function as a public function, as it may bring risks.

Suggestion: Change function visibility to friend function.

Resolution: The client has followed our suggestion and fixed the issue.

LEND-01 Premature Assertion Checks

Severity: Minor

Status: Fixed

Code Location: packages/aries-new/sources/lending/lending.move#L458.

Descriptions: The assertion check at line 458 in `lending.move` can be moved to the top of the function to reduce gas expenditure.

Suggestion: The assertion check at line 458 in `lending.move` can be moved to the top of `lending::deposit_with_repay()` function.

Resolution: The client has followed our suggestion and fixed the issue.

LEND-02 Unverified Amounts Being Set to 0

Severity: Minor

Status: Fixed

Code Location: packages/aries-new/sources/lending/lending.move.

Descriptions: It was observed that the deposit, withdraw, borrow, and repay functions did not include any validation to verify whether the amounts being processed were equal to zero. Even

though transactions could still proceed when the amounts were zero, subsequent operations would be irrelevant.

Suggestion: Add assert validation that can check if the amounts being processed are not zero to mitigate this issue.

Resolution: The client has followed our suggestion and fixed the issue.

LEND-03 Reserve Interest Not Updated in a Timely Manner

Severity: **Medium**
Status: **Fixed**

Code Location: packages/aries-new/sources/lending/lending.move.
Descriptions: Currently, the interest is only updated when the user performs an operation. If a user has a loan from a long time ago and the interest has not been updated for a long time, it may result in delayed updating of the user's asset information. The user's actual assets should be less because they need to pay the interest accrued during that period.

Suggestion: Updating interest in a timely manner ensures that user asset information is correct.

Resolution: The client adopts a solution where assets are periodically refreshed to synchronize the interest calculation. This can be considered as one possible solution.

ORC-01 Unused or Improperly Used Functions and Constants

Severity: **Minor**
Status: **Fixed**

Code Location: packages/aries-new/sources/oracle.move#L8, L11, L14, L17, L31;

packages/aries-new/sources/decimal/u128_math.move#L217;

packages/aries-new/sources/lending/profile/profile.move#L96;

packages/aries-new/sources/lending/reserve/reserve_config.move#L49.
Descriptions: Certain private functions such as `oracle::get_oracle_price()` and `u128_math::approx_eq()` are either unused or only used in test functions. Additionally, there are some empty init functions such as `reserve_config::init()` and `profile::init()`, as well as constants such as `EORACLE_NO_DATA`, `EORACLE_PRICE_STALE`, `EORACLE_PRICES_DIVERGE`, and `U64_MAX` that are not being used.

Suggestion: Unused functions and constants, as well as empty init functions, be removed. If the function is only used by test functions, it should be modified to be a test function.

Resolution: The client has followed our suggestion and fixed the issue.

ORC-02 Missing Validation While Updating Oracle Price

Severity: Medium
Status: Fixed

Code Location: packages/aries-new/sources/oracle.move#L125, L183.
Descriptions: The lack of validation for external prices retrieved during `Oracle` price update using functions `update_oracle_price()` and `update_oracle_price_sb()` can result in a system breakdown in the contract's price system, causing loss of assets for users or the platform. If the external price retrieved equals 0, it should not be used.

Suggestion: Add an `assert` statement to validate that external prices cannot be equal to 0.

Resolution: The client has followed our suggestion and fixed the issue.

DEC-01 Overflow Risk in Utility Functions

Severity: Minor
Status: Fixed

Code Location: packages/aries-new/sources/decimal/decimal.move.
Descriptions: In the `decimal.move` the file, it was found that some multiplication and addition/subtraction operations were not being checked for overflow risk in some functions.

Suggestion: Implement proper bounds checking on all math functions and operations to prevent overflow risks.

Resolution: The client has followed our suggestion and fixed the issue.

RSVST-01 Incorrect Event Timestamp Parameter Setting

Severity: Minor
Status: Fixed

Code Location: packages/aries-new/sources/lending/reserve/reserve_status.move#L505.
Descriptions: It was identified that within the `reserve_status::emit_record_status()`

function, the `record_timestamp` parameter was constantly set to 0 when emitting an event.

Suggestion: `record_timestamp` should be changed to the correct timestamp, `now()`, when emitting an event.

Resolution: The client has followed our suggestion and fixed the issue.

PRF-01 Variable Naming is Not Standardized

Severity: Informational
Status: Acknowledged

Code Location: packages/aries-new/sources/lending/profile/profile.move#L146.
Descriptions: The variable name `address` used in the function `add_profile()` is not compliant with the naming convention.

Suggestion: Rename the variable `address` to a non-keyword name.

PRF-02 The Variable Name Does Not Match its Function

Severity: Informational
Status: Acknowledged

Code Location: packages/aries-new/sources/lending/profile/profile.move#L346.
Descriptions: The function `get_deposit_share_entries()` is intended to retrieve `deposit_share`, but the current variable named `share_arr` does not match the purpose of the function. This inconsistency in variable naming conventions can confuse readers.

Suggestion: Please rename the variable to align with the function's purpose in `get_deposit_share_entries()`.

LQD-01 Missing Market Checks

Severity: Major
Status: Fixed

Code Location: packages/aries-new/sources/lending/liquidate.move#L35.
Descriptions: The lack of validation for the market in the function `liquidate()` may cause the market not to correspond to the current profile.

Suggestion: Call the function `profile::market_match_or_throw()` to ensure validation.

Resolution: The client has followed our suggestion and fixed the issue.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed**: The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an

endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.