MOVEBIT
Securing the Move Ecosystem

Aries Market Contracts **Audit Report**

https://twitter.com/movebit_

contact@movebit.xyz

Aries Market Contracts Audit Report

1 Executive Summary

1.1 Project Information

Type	Lending
Auditors	MoveBit
Timeline	2023–02–13 to 2023–03–21
Languages	Move
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	

1.2 Issue Statistic

Item	Count	Fixed	Pending
Total	12	10	2
Minor	7	6	1
Medium	5	4	1
Major			
Critical			

1.3 Issue Level

- **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.

1.4 Issue Status

- **Fixed:** The issue has been resolved.
- **Pending:** The issue has been acknowledged by the code owner, but has not yet been resolved. The code owner may take action to fix it in the future.

2 Summary of Findings

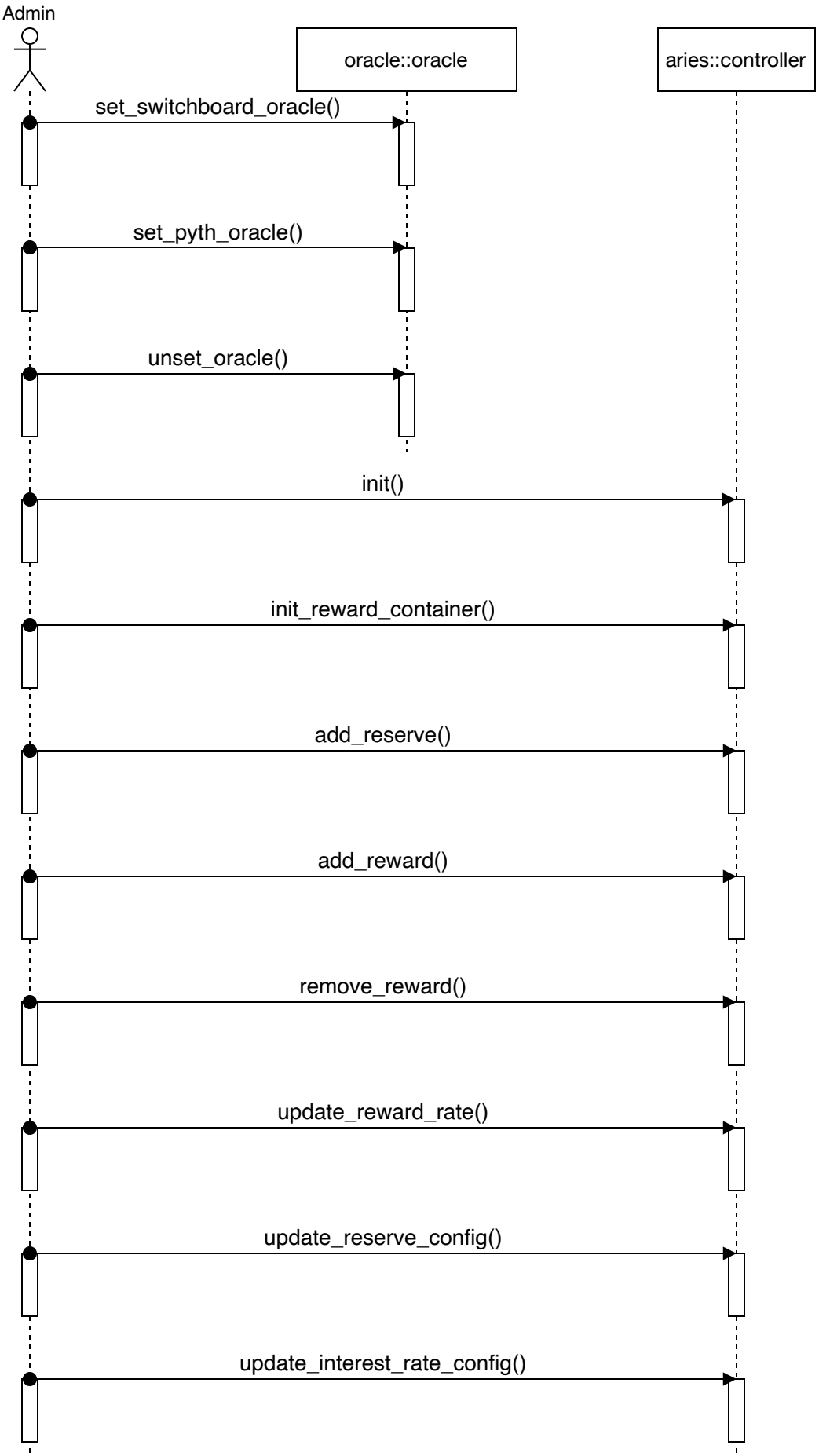
Aries Market is a decentralised margin trading protocol on Aptos. Lend, borrow and trade with margin via a fully on-chain order book with a lightning speed. Our team mainly focused on reviewing the code security and normative, then conducted code running tests and business logic security tests on the test net, our team has been in close contact with the developing team for the past few days. As a result, our team found a total of 12 issues. The team discussed these issues together and communicated with the development team.

3 Participant Process

Here are the relevant actors with their respective abilities within the `Aries-Markets` Smart Contract：

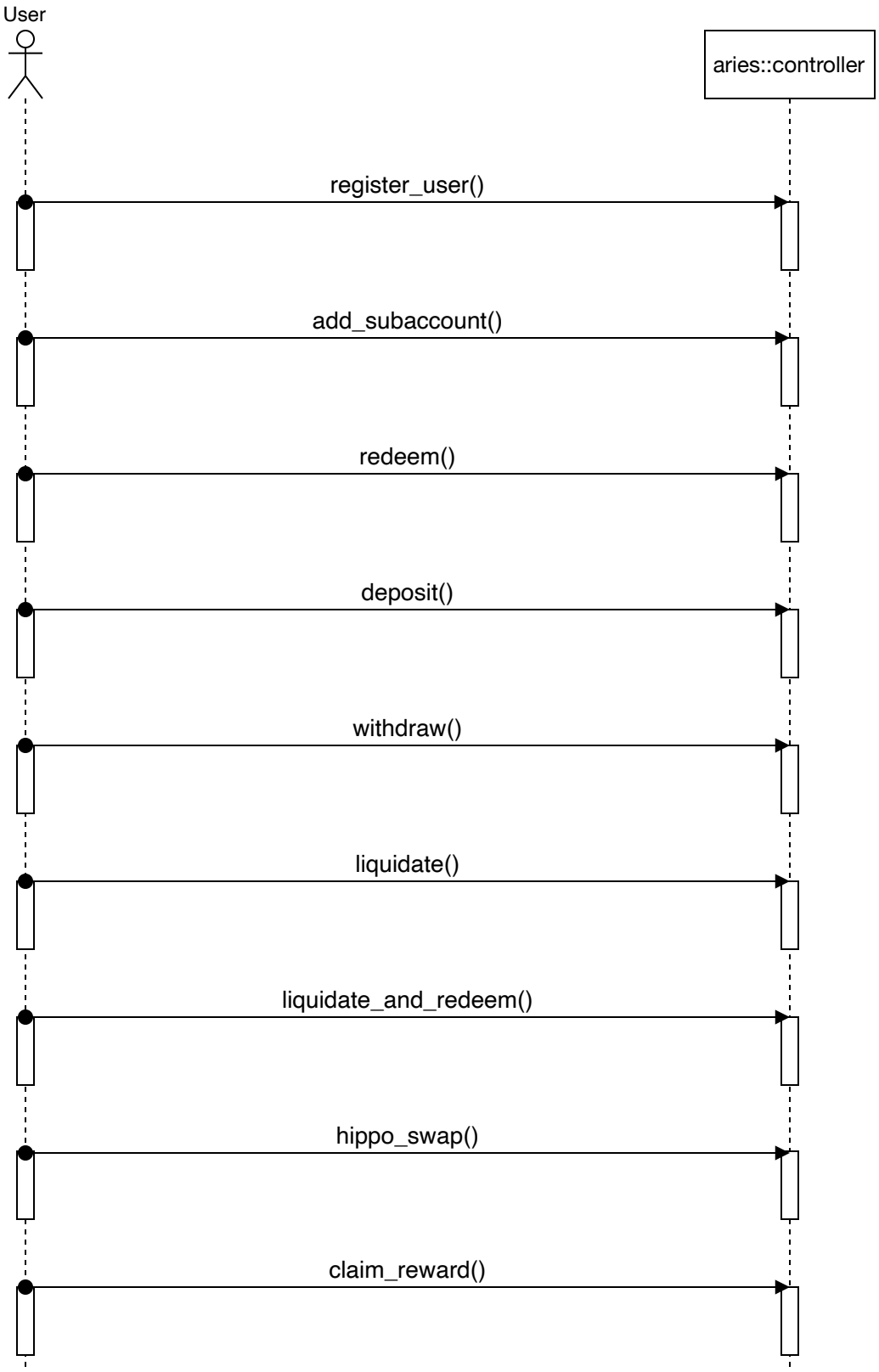
(1) Admin

- Admin to initialize some configuration.
- The administrator initializes the reward of the coin.
- The administrator initializes the coin reserve.
- Admin can add reserve liquidity mining rewards.
- Admin can remove reserve liquidity mining rewards.
- Admin can update reward rate.
- Admin can update reserve config.
- Admin can update rate configuration.
- Admin can set `switchboard oracle` and `pyth oracle`.
- Admin can disable oracle.


Admin
oracle::oracle
aries::controller
set_switchboard_oracle()
set_pyth_oracle()
unset_oracle()
init()
init_reward_container()
add_reserve()
add_reward()
remove_reward()
update_reward_rate()
update_reserve_config()
update_interest_rate_config()

(2) User

- User can register.
- User can add a subaccount.
- User can redeem the yield-bearing LP tokens from a given user.
- User can deposit.
- User can withdraw.
- When the user's loan reaches the liquidation threshold, it can be liquidated by other users.
- User can Leverage-enabled swap.
- User can claim the reward.



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

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", and that 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

Code scope sees **Appendix 1**.

(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.

6 Findings

6.1 There is no assert in the function to verify whether the amount is greater than 0

Severity: Minor

Status: Fixed

Descriptions: It is not verified whether the `amount` is 0 before recharging. According to the code logic, it will be verified when the function `profile::repay_profile` is executed, which undoubtedly consumes excess Gas.

Code Location: packages/aries/sources/controller.move, line 158.

controller.move

```
public entry fun deposit<Coin0>(
    account: &signer,
    profile_name: vector<u8>,
    amount: u64,
    repay_only: bool,
) {
    let addr = signer::address_of(account);
    deposit_for<Coin0>(account, profile_name, amount, addr, repay_only);
}
```

Suggestion: Add an assert at the front of the function to verify that the `amount` must be greater than 0, and modify it as follows:

controller.move

```
public entry fun deposit<Coin0>(
    account: &signer,
    profile_name: vector<u8>,
    amount: u64,
    repay_only: bool,
) {
    assert!(amount > 0, 1);
    let addr = signer::address_of(account);
    deposit_for<Coin0>(account, profile_name, amount, addr, repay_only);
}
```

6.2 Oracle max_deviation can not be updated in `set_pyth_oracle` / `set_switchboard_oracle` function

Severity: Medium

Status: Fixed

Descriptions: The parameter `max_deviation_bips` in `oracle::set_switchboard_oracle` function is only used in the default parameter of `table::borrow_mut_with_default`. Only when the OracleInfo of the coin is not set, the parameter will take effect. The `max_age` and `weight` will be updated if they exist, while `max_deviation_bips` behaves differently. Currently, the only workaround to update the `max_deviation` is `unset_oracle` and set them again.

Code Location: packages/oracle/sources/oracle.move, line 208.

oracle.move

```
public entry fun set_switchboard_oracle<Coin0>(
    account: &signer,
    sb_addr: address,
    max_deviation_bips: u64,
    max_age: u64,
    weight: u64
) acquires OracleIndex {
    let oracle = borrow_global_mut<OracleIndex>(@oracle);
    assert!(signer::address_of(account) == oracle.admin, 0);
    let oracle_info = table::borrow_mut_with_default(
        &mut oracle.prices,
        type_info::type_of<Coin0>(),
        new_oracle_info(coin::decimals<Coin0>(), decimal::from_bips((max_devia
tion_bips as u128))),
    );
    oracle_info.switchboard = option::some(SwitchboardConfig{ sb_addr, max_age
, weight });
    validate_oracle_info(oracle_info);
}
```

Suggestion: Update `max_deviation` of `oracle_info` as well in both `set_switchboard_oracle` and `set_pyth_oracle` functions.

oracle.move

```
public entry fun set_switchboard_oracle<Coin0>(
    account: &signer,
    sb_addr: address,
    max_deviation_bips: u64,
    max_age: u64,
    weight: u64
) acquires OracleIndex {
    let oracle = borrow_global_mut<OracleIndex>(@oracle);
    assert!(signer::address_of(account) == oracle.admin, 0);

    let oracle_info = table::borrow_mut_with_default(
        &mut oracle.prices,
        type_info::type_of<Coin0>(),
        new_oracle_info(coin::decimals<Coin0>(), decimal::from_bips((max_devia
tion_bips as u128))),
    );
    oracle_info.switchboard = option::some(SwitchboardConfig{ sb_addr, max_age
, weight });
    oracle_info.max_deviation = decimal::from_bips((max_deviation_bips as u128
));
    validate_oracle_info(oracle_info);
}
```

6.3 The assert condition is not accurate

Severity: Medium

Status: Fixed

Descriptions: According to the code logic, the function of `assert` here is to prevent addition overflow, but the conditions here are not strict and `U64_MAX * U64_MAX` is not equal to `U128_MAX` .

Code Location: packages/decimal/sources/decimal.move, line 124.

decimal.move

```
public fun add(a: Decimal, b: Decimal): Decimal {
    assert!(a.val + b.val < U64_MAX * U64_MAX, EU128_OVERFLOW);
    Decimal {val: a.val + b.val}
}
```

```
256     const U128_MAX: u128 = 340282366920938463463374607431768211455;
257     #[test]
258     fun test_add2() {
259         assert!(U64_MAX * U64_MAX != U128_MAX, EU128_OVERFLOW);
260     }
261
```

```
$ aptos move test -f test_add2 --ignore-compile-warnings
INCLUDING DEPENDENCY MoveStdlib
BUILDING Decimal
Running Move unit tests
[ PASS    ] 0x3a8dc5ad8fedf897dbcc71eac6e02772510f06d2c46121571b18118a2f06a78f::decimal::test_add2
Test result: OK. Total tests: 1; passed: 1; failed: 0
{
  "Result": "Success"
}
```

Suggestion: Modify the assert condition as follows:

decimal.move

```
const U128_MAX: u128 = 340282366920938463463374607431768211455;
public fun add(a: Decimal, b: Decimal): Decimal {
    assert!(a.val + b.val <= U128_MAX, EU128_OVERFLOW);
    Decimal {val: a.val + b.val}
}
```

6.4 `Key` of generic type `Map` lacks ability constraints `copy + drop`

Severity: Medium

Status: Fixed

Descriptions: The lack of `copy & drop` ability constraints of the generic `Key` of the structure `Map` causes compilation errors.

Code Location: packages/util-types/sources/map.move, line 19.

map.move

```
struct Map<Key, Value> has copy, drop, store {
    data: vector<Element<Key, Value>>,
}
```

```
error[E05001]: ability constraint not satisfied
   ┌─
19 │     struct Map<Key, Value> has copy, drop, store {
   │                ------------ To satisfy the constraint, the 'copy' ability would need to be added here
20 │         data: vector<Element<Key, Value>>,
   │                      ^^^^^^^^^^^^^^^^^^^^
   │                      │       │
   │                      │       The type 'Key' does not have the ability 'copy'
   │                      'copy' constraint not satisifed
   ·
24 │     struct Element<Key: copy + drop, Value> has copy, drop, store {
   │                         ---- 'copy' constraint declared here

error[E05001]: ability constraint not satisfied
   ┌─
19 │     struct Map<Key, Value> has copy, drop, store {
   │                ------------ To satisfy the constraint, the 'drop' ability would need to be added here
20 │         data: vector<Element<Key, Value>>,
   │                      ^^^^^^^^^^^^^^^^^^^^
   │                      │       │
   │                      │       The type 'Key' does not have the ability 'drop'
   │                      'drop' constraint not satisifed
   ·
24 │     struct Element<Key: copy + drop, Value> has copy, drop, store {
   │                           ---- 'drop' constraint declared here

{
  "Error": "Move compilation failed: Compilation error"
}
```

Suggestion: Add `copy + drop` ability constraints to the generic `Key` of the structure `Map`, modify as follows:

map.move

```
struct Map<Key: copy + drop, Value> has copy, drop, store {
    data: vector<Element<Key, Value>>,
}
```

6.5 Package upgrade policy risk

Severity: Medium

Status: Pending

Descriptions: The current packages are using the default upgrade policy `compatible`, which enables the deployer to add new functions. Since all the assets in the product are kept in `ReserveCoinContainer` struct under the deployer's namespace. The deployer can upgrade the

package, add a new function to take all the coins out of `ReserveCoinContainer`, and transfer them to other addresses. The users' assets will be in danger.

Suggestion: After the packages are stable enough, changing the upgrade policy to `immutable` will ensure that the packages are much safer for users.

Move.toml

```
[package]
name = "Aries"
version = "0.0.0"
upgrade_policy = "immutable"
```

If the upgrade is still necessary for further development, it's better to use a multi-signature account as the deployer or governance solution for the upgrade.

6.6 The code specification is not uniform

Severity: Minor

Status: Fixed

Descriptions: The calling methods of the two `iterable_table::contains` in the function `remove_collateral_profile` are inconsistent, one of which lacks the generic parameters, and the error code of assert should be defined as a constant `EPROFILE_NO_BORROWED_RESERVE`.

Code Location: packages/aries/sources/profile.move, line 476.

profile.move

```
fun remove_collateral_profile(
    profile: &mut Profile,
    reserve_type_info: TypeInfo,
    amount: u64
    ): u128 {
    assert!(!iterable_table::contains(&profile.borrowed_reserves, reserve_type
_info), 0);
    assert!(
        iterable_table::contains<TypeInfo, Deposit>(
            &profile.deposited_reserves, reserve_type_info
        ),
        EPROFILE_NO_DEPOSIT_RESERVE
    );
    ......
}
```

Suggestion: Add the generic parameters and modify the error code to be a constant, as follows:

profile.move

```
fun remove_collateral_profile(
    profile: &mut Profile,
    reserve_type_info: TypeInfo,
    amount: u64
    ): u128 {
    assert!(!iterable_table::contains<TypeInfo, Reserves>(&profile.borrowed_re
serves, reserve_type_info), EPROFILE_NO_BORROWED_RESERVE);
    assert!(
        iterable_table::contains<TypeInfo, Deposit>(
            &profile.deposited_reserves, reserve_type_info
        ),
        EPROFILE_NO_DEPOSIT_RESERVE
    );
    ......
}
```

6.7 The project can not run unit test with latest aptos cli(v1.0.6)

Severity: Minor

Status: Pending

Descriptions: The latest aptos cli(v1.0.6) forces the unit test failure macro # `[expected_failure(abort_code = 1)]` to have a location parameter. The location parameter is used to specify the location of the failure. It's better to make it compatible and testable with the latest aptos cli for further maintenance.

Code Location: packages/aries, packages/aries-config, packages/decimal, packages/free-coin.

Suggestion: Fix as below example. Most of the `location` can be Self, and some of them may have to be specified manually.

```
#[expected_failure(location = Self, abort_code = 1)]
```

6.8 Bad validation condition for function caller

Severity: Medium

Status: Fixed

Descriptions: The admin address is set in `controller::init`. The address here can be any address. When calling `controller::add_reserve()`, the first line in the function `add_reserve` verifies that it must be called by admin, and then call `reserve::create`, The assert condition in `reserve::create` is that the caller must be `@aries`, which is inconsistent with the assert condition in `controller::add_reserve`, and also violates the original intention of `controller::init` to set the administrator.

Assuming that an address other than `@aries` is set when calling `controller::init`, then calling `controller::add_reserve` will definitely abort, and this function will never be successfully called. There is no resource for creating this coin, and the entire subsequent process will be aborted. Unable to proceed.

Code Location: packages/aries/sources/reserve.move, line 122.

controller.move

```
public entry fun init(account: &signer, admin_addr: address) {
    controller_config::init_config(account, admin_addr);
    reserve::init(account);
    oracle::init(account, admin_addr)
}
```

controller.move

```
public entry fun add_reserve<Coin0>(admin: &signer) {
    controller_config::assert_is_admin(signer::address_of(admin));
    // TODO: change to use `lp_store` since it won't always be the same as th
e `admin`.
    // TODO: change to custom configuration.
    reserve::create<Coin0>(
        admin,
        decimal::one(),
        reserve_config::default_config(),
        interest_rate_config::default_config()
    )
}
```

reserve.move

```
public(friend) fun create<Coin0>(
    account: &signer,
    initial_exchange_rate: Decimal,
    reserve_config: ReserveConfig,
    interest_rate_config: InterestRateConfig
) acquires Reserves {
    assert!(signer::address_of(account) == @aries, ERESERVE_NOT_ARIES);

    ......
}
```

Suggestion: Modify the assert condition in the function `reserve::create` as follows:

reserve.move

```
public(friend) fun create<Coin0>(
    account: &signer,
    initial_exchange_rate: Decimal,
    reserve_config: ReserveConfig,
    interest_rate_config: InterestRateConfig
) acquires Reserves {
    controller_config::assert_is_admin(signer::address_of(account));

    ......
}
```

6.9 The assert judgment condition is inaccurate

Severity: Minor

Status: Fixed

Descriptions: In the function `validate_oracle_info`, when `sb_weight=0`, `pyth_weight=U64_MAX`, or `sb_weight=U64_MAX`, `pyth_weight=0`, the assertion condition is not met, causing oracle verification to fail.

Code Location: packages/oracle/sources/oracle.move, line 95.

oracle.move

```
fun validate_oracle_info(info: &OracleInfo) {
    ......
    if (option::is_some(&info.switchboard) && option::is_some(&info.pyth)) {
        let sb_weight = option::borrow(&info.switchboard).weight;
        let pyth_weight = option::borrow(&info.pyth).weight;
        assert!(sb_weight + pyth_weight > 0 && sb_weight < U64_MAX - pyth_weig
ht, 0);
    }
}
```

Suggestion: Modify the assert condition as follows:

oracle.move

```
fun validate_oracle_info(info: &OracleInfo) {
    ......
    if (option::is_some(&info.switchboard) && option::is_some(&info.pyth)) {
        let sb_weight = option::borrow(&info.switchboard).weight;
        let pyth_weight = option::borrow(&info.pyth).weight;
        assert!(sb_weight + pyth_weight > 0 && sb_weight <= U64_MAX - pyth_wei
ght, 0);
    }
}
```

6.10 Inaccurate judgment on deposit and loan restrictions

Severity: Minor

Status: Fixed

Descriptions: In the following functions, when determining whether the deposit and loan amount exceeds the limit, the numerical comparison is inaccurate, and the numerical value should be less than or equal to the limit, not less than.

Code Location: packages/aries/sources/reserve_details.move , line 172,177.

reserve_details.move

```
fun is_within_deposit_limit(reserve_details: &ReserveDetails): bool {
    ......
    (total_liquidity as u64) < deposit_limit
}

fun is_within_borrow_limit(reserve_details: &ReserveDetails): bool {
    ......
    decimal::as_u64(reserve_details.total_borrowed) < borrow_limit
}
```

Suggestion: Modify as follows:

reserve_details.move

```
fun is_within_deposit_limit(reserve_details: &ReserveDetails): bool {
    ......
    (total_liquidity as u64) <= deposit_limit
}

fun is_within_borrow_limit(reserve_details: &ReserveDetails): bool {
    ......
    decimal::as_u64(reserve_details.total_borrowed) <= borrow_limit
}
```

6.11 The function `init_reward_container` is missing caller constraints

Severity: Minor

Status: Fixed

Descriptions: The function `init_reward_container` should only allow `@aries` calls, but there is no restriction here.

Code Location: packages/aries/sources/controller.move , line 38.

controller.move

```
public entry fun init_reward_container<Coin0>(account: &signer) {
    reward_container::init_container<Coin0>(account);
}
```

Suggestion: Add an assert to restrict the caller to only `@aries` :

controller.move

```
public entry fun init_reward_container<Coin0>(account: &signer) {
    assert!(signer::address_of(account) == @aries, 0);
    reward_container::init_container<Coin0>(account);
}
```

6.12 Assertion error code is incorrect

Severity: Minor

Status: Fixed

Descriptions: There is an assertion in the function `profile::new` whether `addr` owns Profiles, but the returned error code is `EPROFILE_ALREADY_EXIST` , which should be `EPROFILE_NOT_EXIST` .

Code Location: packages/aries/sources/profile.move , line 130.

profile.move

```
public fun new(account: &signer, profile_name: string::String) acquires Profiles {
    let addr = signer::address_of(account);
    assert!(exists<Profiles>(addr), EPROFILE_ALREADY_EXIST);
    ......
}
```

Suggestion: Replace `EPROFILE_ALREADY_EXIST` with `EPROFILE_NOT_EXIST` .

controller.move

```
public fun new(account: &signer, profile_name: string::String) acquires Profiles {
    let addr = signer::address_of(account);
    assert!(exists<Profiles>(addr), EPROFILE_NOT_EXIST);
    ......
}
```

Appendix 1 – Files in Scope

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

Files	SHA-1 Hash
decimal/sources/decimal.move	14a2d0014d7ced94ff9d47604943bcf6badab9c8
decimal/sources/u128_math.move	cef4c7fa9265012433a098cd65d355be3f0cf207
decimal/sources/modules.spec.move	258c12c9f376b351ca3476e7e9ea61aa65ded9b9
decimal/sources/uint.move	ade1f3dfc0f6ad41f1b39881a884d2f9072fefe7
oracle/sources/oracle.move	095a40b0630f8ff07493cfb672ee194985e1d756
aries-config/sources/interest_rate_config.move	ad3f5e3e012c3bc323ae9f366f419cd1ac59c5c5
aries-config/sources/reserve_config.move	fec13c1371f6b43be476f772c86a9f3a52ed1107
free-coin/sources/free_coin.move	ded53db99fa4fbcf129281980d027d41cff53860
aries/sources/utils.move	5e7e0b65c66f5489a12586bbfb2523c8df3789ed
aries/sources/referral.move	0c0d8bb5162ea45f837c245e041d183a792d4a84
aries/sources/controller_config.move	1938e4ebca11939260ec0257bd74526760769fd8
aries/sources/profile.move	a9b4ba7253ac40f074c8b9754d5f540d2e6074b2
aries/sources/reserve_farm.move	d776359d213feb9e915f43ac8b5c58f52df5e00b
aries/sources/profile_farm.move	4e8c684b0ec0d01e2e618d6b64b6426da16a9dde
aries/sources/modules.spec.move	6d88a7b23f6d5f08e75ea462270b87cdd0c36cde
aries/sources/reserve.move	c30fca10a535b913dfd6ae99d4dd26a8111c1b64
aries/sources/reserve_details.move	c88352c973e3d6ecb5740759def9107ab553244b

aries/sources/reward_container.move	2e29c7d6130fa08ef7e4c5803539ac7794591a0f
aries/sources/math_utils.move	7babf980e0ff23131890f9162da0166fedd8df46
aries/sources/controller.move	c4322f499c78e5097832bbc3dfd2be18f958fe1d
aries/sources/borrow_type.move	40ce61aaee69e617e523048db8a497bcbf371e83
aries-wrapper/sources/wrapped_controller.move	bf09a1ab4ce41193ad87862055cd2150bba0f1b1
util-types/sources/pair.move	888667d11f997cec148d4ecc442123def41a7489
util-types/sources/map.move	69c94f591446d5121104132a5678abf74c8c1f3e
util-types/sources/iterable_table.move	0293e78965eb9121f3c4971e974d1e83652d2c3b

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.

MOVEBIT

Securing the Move Ecosystem

https://twitter.com/movebit_

contact@movebit.xyz