Amnis Finance

Audit Report

contact@movebit.xyz

https://twitter.com/movebit_

Fri Oct 13 2023

Amnis Finance Audit Report

1 Executive Summary

1.1 Project Information

Description	A Pioneering Liquidity Staking on Aptos
Type	Staking
Auditors	MoveBit
Timeline	Wed Sep 20 2023 - Fri Oct 13 2023
Languages	Move
Platform	Aptos
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/amnis-finance/amnis-contract
Commits	3dcc48845e5ec26ef34736d8eee72af8d2fe779e 59525ac8e9c16904b675c740c3df5017088ba637 9fec843722993d201b02439f6fc0446aa0c169fc 0fa22c304b3c54d3759fa86ad3f6784189bf48c8

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MOV	Move.toml	9dbd0500e959d81c973d7fc9c298ed0b0a2cd891
AGO	sources/aptos_governance.move	fb2b3b8459aa902196eb4ea65ec8729ec6668698
PMA	sources/package_manager.move	4a91dc11f9f86305c2f25778f15a065f138aa791
ROU	sources/router.move	1b1c20b6c0e703bd3d4ac9fd63c4d10604ac0772
WIT	sources/withdrawal.move	d13342d94661889093d6a75b93189403fee18f23
ATO	sources/amapt_token.move	23e6c6090bac4a1cbc139c6871145238cb4e76cd
TRE	sources/treasury.move	814359ea7b2d3afafd26d378ad8ac91f14c058e0
GOV	sources/governance.move	3188f88b5399ffbf261dbf5e326670d576c2967b
STO	sources/stapt_token.move	87e606d24754a50e8c13c34d5e9ef0c87fe9978a
DMA	sources/delegation_manager.move	f49de208266c4e6a8e06847738c24b1d8593a4e2

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	6	5	1
Informational	0	0	0
Minor	3	3	0
Medium	2	2	0
Major	1	0	1
Critical	0	0	0

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 is illustrated 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 Amnis Finance to identify any potential issues and vulnerabilities in the source code of the Amnis Finance 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 identified 6 issues of varying severity, listed below.

ID	Title	Severity	Status
DMA-1	Missing Validation for Returning A Value of 0 May Lead to Computational Overflow	Medium	Fixed
DMA-2	Unused Constant	Minor	Fixed
GOV-1	Centralization Risk	Major	Acknowledged
GOV-2	Lack of Invocation Permissions for Initialization	Minor	Fixed
ROU-1	The `MIN_AMOUNT` Deposit Cannot Be Withdrawn	Minor	Fixed
STO-1	Permission Issues Related to Adding Tokens	Medium	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the Amnis Finance Smart Contract:

User

- Users can lock their amAPT tokens for Aptos on-chain governance voting using the `lock_amapt` function.
- Users can lock their stAPT tokens for Aptos on-chain governance voting using the `lock_stapt` function.
- Users can lock their tokens until a proposal expires and vote on the given proposal with the `vote_on` function.
- Users can unlock their tokens once the lockup period has expired using the `unlock` function.
- Users can deposit APT and receive amAPT tokens using the `deposit_entry` function.
- Users can stake amAPT tokens to receive stAPT tokens using the `stake_entry` function.
- The `unstake_entry` function permits users to unstake stAPT tokens to receive amAPT tokens.
- Users can request the withdrawal of amAPT tokens into APT tokens using the `request_withdrawal_entry` function.
- The `withdraw_entry` function enables users to withdraw tokens into their account.
- The `withdraw_multi_entry` function allows users to withdraw multiple tokens into their account.
- Users can invoke the `update_rewards` function once per epoch to refresh the rewards for all validators.

Admin

- The `initialize` function allows the admin to initialize various components.
- The `whitelist_stake_pool` function enables the admin to whitelist a stake pool for delegation, depositing an initial amount of APT into the newly whitelisted pool if required.
- The `suspend_stake_pool` function allows the admin to suspend a stake pool, preventing further delegation to it.

- The `update_treasury_fee` function permits the admin to update the treasury fee in basis points (bps).
- The `set_withdrawal_lockup_duration` function enables the admin to set the withdrawal lockup duration.
- The `maintain_treasury_amapt_reserves` function allows the admin to maintain liquidity reserves in the treasury.
- The `upgrade` function enables the admin to upgrade the system by providing new package metadata and code.

4 Findings

DMA-1 Missing Validation for Returning A Value of 0 May Lead to Computational Overflow

Severity: Medium

Status: Fixed

Code Location:

sources/delegation_manager.move#L309-322

Descriptions:

The primary purpose of the function `update_reward()` is to calculate the performance score based on the delegator's rewards and delegation status and update this score in the delegation record.The performance score is calculated based on the delegator's stake and the reward rate, with the reward rate obtained from the `staking_config::get_reward_rate()` function.

```
 let (reward_rate_num, reward_rate_denom) =
staking_config::get_reward_rate(&staking_config::get());
        let reward_rate_num = (reward_rate_num as u128);
        let reward_rate_denom = (reward_rate_denom as u128);
        let max_reward = math128::mul_div(prev_working_stake, reward_rate_num,
reward_rate_denom);
        // Deduct one to account for rounding error. We'll make sure the performance
score is never > 1 later.
        max_reward = max_reward * (BPS_MAX - commission_rate_bps) / BPS_MAX - 1;
        let actual_rewards = total_stake(&curr_state) - total_stake(&prev_state);
        actual_rewards * BPS_MAX / max_reward
```

As shown below the function `staking_config::get_reward_rate()` can potentially return a 0 value for `reward_rate_num` .

```
  public fun get_reward_rate(config: &StakingConfig): (u64, u64) acquires
StakingRewardsConfig {
      if (features::periodical_reward_rate_decrease_enabled()) {
        let epoch_rewards_rate = borrow_global<StakingRewardsConfig>
(@aptos_framework).rewards_rate;
        if (fixed_point64::is_zero(epoch_rewards_rate)) {
```

```
            (0u64, 1u64)
        } else {
            // Maximize denominator for higher precision.
            // Restriction: nominator <= MAX_REWARDS_RATE && denominator <= MAX_U64
            let denominator = fixed_point64::divide_u128((MAX_REWARDS_RATE as u128),
epoch_rewards_rate);
            if (denominator > MAX_U64) {
                denominator = MAX_U64
            };
            let nominator = (fixed_point64::multiply_u128(denominator,
epoch_rewards_rate) as u64);
            (nominator, (denominator as u64))
        }
    } else {
        (config.rewards_rate, config.rewards_rate_denominator)
    }
}
```

When `reward_rate_num` is 0, the calculation of `max_reward` will result in 0, division by zero occurs in the calculation `actual_rewards * BPS_MAX / max_reward`.

Suggestion:

It is recommended to ensure that `max_reward` is never zero or that division by zero.

Resolution:

This issue has been fixed. The client added the additional check.

DMA-2 Unused Constant

Severity: Minor

Status: Fixed

Code Location:

sources/delegation_manager.move#L33

Descriptions:

The main consequence of the Unused Constants defect is the increase in gas costs during module deployment, leading to gas wastage.

```
const MAX_U64: u64 = 18446744073709551615;
```

Suggestion:

It is recommended to delete unused constants.

Resolution:

This issue has been fixed. The client has used the constants.

GOV-1 Centralization Risk

Severity: Major

Status: Acknowledged

Code Location:

sources/governance.move#L19-22

Descriptions:

In the `governance` module, there is an issue of excessive centralization in admin permissions, including the ability to execute critical operations such as `whitelist_stake_pool` and `update_treasury_fee`. However, the process of configuring these crucial protocol parameters lacks specific timing or well-defined ranges. This absence of constraints could potentially lead to suboptimal user experiences and introduce risks associated with centralization.

Suggestion:

It is recommended to mitigate smart contract centralization risks by employing the following methods:

- Foster community governance and participation to ensure decision-making power is distributed among the system's participants.
- Implement robust security measures to protect against potential attacks or malicious actions, such as multi-signature.
- Introducing a delay when configuring critical parameters is an effective security measure that can provide additional protection for the decentralization and security of the system.

GOV-2 Lack of Invocation Permissions for Initialization

Severity: Minor

Status: Fixed

Code Location:

sources/governance.move#L24

Descriptions:

If a bad actor calls the initialization function of another module, such as `amapt_token::initialize()` or `stapt_token::initialize()`, before the `governance::initialize()` function is invoked, it can disrupt the intended initialization process.

Suggestion:

It is recommended that `amapt_token::initialize()` and `stapt_token::initialize()` should not be callable independently and should only be executed through friend functions.

Resolution:

This issue has been fixed. The client added friend functions.

ROU-1 The `MIN_AMOUNT` Deposit Cannot Be Withdrawn

Severity: Minor

Status: Fixed

Code Location:

sources/router.move#L138-145

Descriptions:

In the `deposit` function, users deposit APT to receive amAPT, and there is a minimum value check (`MIN_AMOUNT`) in place. If a user deposits exactly the minimum amount of APT, which is 0.1 APT, fees are deducted, resulting in the user minting amAPT for an amount less than 0.1 APT.

```
  /// Deposit APT to receive amAPT.
  public fun deposit(apt: Coin<AptosCoin>): Coin<AmnisApt> {
    // Don't need to call update_rewards here delegation_manager will do so.
    let apt_amount = coin::value(&apt);
    assert!(apt_amount >= MIN_AMOUNT, EINSUFFICIENT_AMOUNT);
    let add_stake_fee = delegation_manager::add_stake(apt);
    // Deduct add stake fee charged by delegation_pool and only min the equivalent
amAPT amount after fees.
    amapt_token::mint(apt_amount - add_stake_fee)
  }
```

This situation prevents users from passing the minimum value check when attempting to request redemption of amAPT for APT through the `request_withdraw` function.

```
assert!(amount >= MIN_AMOUNT, EINSUFFICIENT_AMOUNT);
```

Suggestion:

It is recommended that in the `deposit` function, the check should be on the received amount of amAPT being greater than `MIN_AMOUNT` , rather than the amount of APT being deposited.

Resolution:

This issue has been fixed. The client has increased the minimum deposit amount.

STO-1 Permission Issues Related to Adding Tokens

Severity: Medium

Status: Fixed

Code Location:

sources/stapt_token.move#L103

Descriptions:

Regarding the public access function `add()`, which allows any user to add `amAPT` to the `StakedAptManagement` resource. As indicated by the comments, this function should only be called by the `delegation_manager` and `router` modules.

Suggestion:

It is recommended to use the friend function to restrict access permissions.

Resolution:

This issue has been fixed. The client added friend functions.

5 Prover Formal Verification

The formal verification report of some files and modules is as follows.

aptos_governance

General Descriptions

This module is located in `sources/aptos_governance.move`.

This module provides users with the ability to vote for proposals and also includes the functionality of locking and releasing tokens.

Formally Verified Properties

- Make sure that the user has the resource AptosGovernanceParticipation before calculating their voting weight.
- Ensure that the voting weight is determined by the sum of the user's locked `amAPT` and `stAPT`.
- Verify that the amount of locked tokens is greater than zero before proceeding.
- Verify that the amount of locked tokens after the lock-up period is not less than the quantity locked before.
- To ensure that the user no longer has the resource `AptosGovernanceParticipation` after unlocking the lock-up.

governance

General Descriptions

This module is located in `sources/governance.move`.

This module is primarily designed for project parties, and its main functions include project initialization, updating account addresses for various roles, adding whitelists, voting, modifying token lock-up periods, modifying reward fee ratios, etc.

Formally Verified Properties

- To verify that the project has been initialized and the address `@amnis` has the resource `AmnisGovernance` before executing certain operations
- To ensure that the caller remains an administrator after the `transfer_admin()` function is called to transfer administrator privileges, and to verify that the new administrator address is the newly passed-in value `new_admin` .

```
spec transfer_admin(admin: &signer, new_admin: address) {
    let governance = global<AmnisGovernance>(@amnis);
    let post governance_post = global<AmnisGovernance>(@amnis);
    aborts_if !exists<AmnisGovernance>(@amnis);
    aborts_if signer::address_of(admin) != governance.admin;
    ensures governance_post.pending_admin == new_admin;
    ensures governance_post.admin == signer::address_of(admin);
}
```

- To verify that the current caller is the expected address waiting to receive the administrator role when a user accepts administrator privileges through the `accept_admin()` function, and to ensure that after the execution, the `pending_admin` is set to zero address and the caller becomes the new administrator.

```
spec accept_admin(pending_admin: &signer) {
    let governance = global<AmnisGovernance>(@amnis);
    let post governance_post = global<AmnisGovernance>(@amnis);
    aborts_if !exists<AmnisGovernance>(@amnis);
    aborts_if signer::address_of(pending_admin) != governance.pending_admin;
    ensures governance_post.pending_admin == @0x0;
    ensures governance_post.admin == governance.pending_admin;
}
```

- To verify that the caller of `set_treasury_operator()` can only be the administrator and ensure that after execution, the `treasury_operator` has been changed to a new address.
- To verify that the caller of `set_whitelistor()` can only be the administrator and ensure that after execution, the whitelistor has been changed to a new address.
- Ensure that only administrators can modify `treasury_fee_bps` and that the modified value matches the expected value set by the administrator.

stapt_token

General Descriptions

This module is located in `sources/stapt_token.move`.

This is the `stAPT` module for the platform coin released by the project party. Users can obtain `stAPT` by staking `amAPT`. This module includes functions such as initializing tokens, retrieving token prices, depositing and minting `stAPT`, and `unstake` and burning `stAPT`.

Formally Verified Properties

- Verify that the address `@amnis` does not have the resource `StakedAptManagement` before initialization, and that `PermissionConfig` already exists.
- Ensure that the address `@amnis` has the resource `StakedAptManagement` after initialization and that other function calls occur after initialization.
- Verify overflow when calculating `stAPT` prices.
- If the total supply of `stAPT` is 0, the ratio between `stAPT` and `amAPT` is 1:1. Otherwise, the price calculation formula for `stAPT` is `staked_apt_management.amapt * PRECISION / total_stapt_supply`.

```
spec stapt_price(): u64 {
    let total_stapt_supply = spec_total_supply();
    let staked_apt_management = global<StakedAptManagement>(@amnis);
    aborts_if total_stapt_supply > 0 && coin::value(staked_apt_management.amapt) *
PRECISION / total_stapt_supply > MAX_U64;

    ensures result == stapt_price_result();
}

spec fun stapt_price_result(): u64 {
    let total_stapt_supply = spec_total_supply();
    let staked_apt_management = global<StakedAptManagement>(@amnis);
    if (total_stapt_supply == 0) {
        PRECISION
    } else {
        coin::value(staked_apt_management.amapt) * PRECISION / total_stapt_supply
    }
}
```

- The amount of `stAPT` obtained by depositing a certain quantity of `amAPT` meets expectations.

```
spec deposit(amapt: Coin<AmnisApt>): Coin<StakedApt> {
    let staked_apt_management = global<StakedAptManagement>(@amnis);
    let post staked_apt_management_post = global<StakedAptManagement>(@amnis);
    ensures coin::value(staked_apt_management_post.amapt) ==
coin::value(staked_apt_management.amapt) + coin::value(amapt);

    let stapt_price = stapt_price_result();
    let equivalent_stapt = coin::value(amapt) * PRECISION / stapt_price;
    ensures coin::value(result) == equivalent_stapt;
}
```

- When adding `amAPT` to the resource through `add()`, ensure that the balance of `amAPT` after the addition meets expectations.

```
spec add(amapt: Coin<AmnisApt>) {
    let amapt_coin = global<StakedAptManagement>(@amnis).amapt;
    let post amapt_coin_post = global<StakedAptManagement>(@amnis).amapt;
    ensures coin::value(amapt_coin_post) == coin::value(amapt_coin) + coin::value(amapt);
}
```

- When releasing `stAPT` through `unstake()`, ensure that the amount of `amAPT` redeemed matches the expectations and that the remaining `amAPT` has decreased by the amount redeemed.

```
spec unstake(stapt: Coin<StakedApt>): Coin<AmnisApt> {
    aborts_if coin::value(stapt) == 0;
    let stapt_price = stapt_price_result();
    aborts_if coin::value(stapt) * stapt_price > MAX_U128;

    let equivalent_amapt = coin::value(stapt) * stapt_price / PRECISION;
    ensures coin::value(result) == equivalent_amapt;
    let staked_apt_management = global<StakedAptManagement>(@amnis);
    let post staked_apt_management_post = global<StakedAptManagement>(@amnis);
    ensures coin::value(staked_apt_management.amapt) ==
coin::value(staked_apt_management_post.amapt) + equivalent_amapt;
}
```

treasury

General Descriptions

This module is located in `sources/treasury.move`.

This module provides the project party with the functionality to update the reward rate and collect/withdraw reward fees.

Formally Verified Properties

- Ensure that the updated value of `treasury_fee_bps` is the latest value.

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.