Amnis

Audit Report

MOVEBIT

contact@movebit.xyz

https://twitter.com/movebit_

Thu Mar 14 2024

Amnis Audit Report

1 Executive Summary

1.1 Project Information

Description	A Pioneering Liquidity Staking on Aptos
Type	Staking
Auditors	MoveBit
Timeline	Wed Jan 31 2024 - Thu Mar 14 2024
Languages	Move
Platform	Aptos
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/amnis-finance/amnis-contract https://github.com/tony001242/amnis-contract
Commits	https://github.com/amnis-finance/amnis-contract: 08239f04ed57a9642b366f36e5184e727acc3afe https://github.com/tony001242/amnis-contract: 7eb9708269cd298867e668018dbc46362f05e34b

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
ATO	sources/amapt_token.move	23e6c6090bac4a1cbc139c6871145238cb4e76cd
STO	sources/stapt_token.move	87e606d24754a50e8c13c34d5e9ef0c87fe9978a
WIT	sources/withdrawal.move	0de818ee1a794f2c2724e1cf76153e014ebaf290
PMA	sources/package_manager.move	c4747144fa3297384b2a3ac575043974dc82241b
ROU	sources/router.move	e9d2d9f6852d02796c7a1ba55900e43e0bf44b76
DMA	sources/delegation_manager.move	1179adbc9394dc668dabeff70f76e708cbd182c3
PEG	sources/pegging.move	af46ac331cbff9c844baf0d9dee0b51c7ed95d9b
TRE	sources/treasury.move	8f14890b9abd47258b0233edfc7847e7ff70fc57
GOV	sources/governance.move	4b28bf11d85e49859a2eefae307ae14de7136af0

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	3	3	0
Informational	0	0	0
Minor	2	2	0
Medium	1	1	0
Major	0	0	0
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 to identify any potential issues and vulnerabilities in the source code of the Amnis 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 3 issues of varying severity, listed below.

ID	Title	Severity	Status
PEG-1	The `operator` can Evade The Fees When Loaning Assets	Medium	Fixed
PEG-2	Underutilized Constant `EID_INVALID` in `Pegging` Module	Minor	Fixed
PEG-3	Contract Configuration and Loan Validation Improvements	Minor	Fixed

3 Participant Process

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

Gov

- Gov can set the operator through `set_pegging_operator()` .

Treasury operator

- Treasury operator can update `treasury_incentives` through `update_reward_incentives()` .

User

- Users can cancel withdrawals through `cancel_withdraw_multi()` .

operator

- The operator can update the maximum reserve amount through `config_pegging()` .
- The operator can conduct flash loans through `loan_apt()` and `repay_amapt()` .

4 Findings

PEG-1 The `operator` can Evade The Fees When Loaning Assets

Severity: Medium

Status: Fixed

Code Location:

sources/pegging.move#77

Descriptions:

The function `pegging.loan_apt()` allows the operator to withdraw funds from the protocol, but a certain fee is required when returning the funds. The fee calculation is as follows:

```
math64::mul_div(amount, pegging().loan_fee, BPS_MAX)
```

According to the protocol configuration, we found that loan_fee is 10, and `BPS_MAX` is 10000. When `amount * 10 < 10000`, users will not have to pay any fees. Therefore, the operator can repeatedly borrow 999 to avoid the fees.

Suggestion:

It is recommended to set a minimum loan amount or to check if the fee is 0, in which case borrowing assets should not be allowed.

Resolution:

This issue has been fixed by adding a check for `fee > 0` in the protocol.

PEG-2 Underutilized Constant `EID_INVALID` in `Pegging` Module

Severity: Minor

Status: Fixed

Code Location:

sources/pegging.move#19

Descriptions:

The constant `EID_INVALID` in the `pegging` module is not utilized, potentially impacting code readability and causing unnecessary gas consumption.

```
const EID_INVALID: u64 = 3;
```

Suggestion:

It is recommended to remove unused constants.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

PEG-3 Contract Configuration and Loan Validation Improvements

Severity: Minor

Status: Fixed

Code Location:

sources/pegging.move#53,75

Descriptions:

Currently, there is no place in the contract to modify `loan_fee`. Should the ability to update `loan_fee` be allowed in the `config_pegging()` function? Additionally, the `loan_apt()` function checks that the treasury balance must be greater than the loan amount. Should it also allow equality, as a borrower might acquire the entire balance before invoking this function? The current validation may lead to confusion for borrowers attempting to loan their entire balance.

Suggestion:

It is recommended to improve based on the description.

Resolution:

This issue has been fixed. The client has allowed modification of `loan_fee` in `config_pegging()` and has altered the validation conditions in `loan_apt()`.

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.