MOVEBIT
Securing the Move Ecosystem

Mobius NFT AMM
Audit Report


https://twitter.com/movebit_


contact@movebit.xyz

Mobius NFT AMM Audit Report

1 Executive Summary

1.1 Project Information

Type	NFT-Fi
Auditors	MoveBit
Timeline	2023-02-27 to 2023-03-09
Languages	Move
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	Repository: https://github.com/MOBIUS-MARKET/mobius-nft-amm Received Commit: 0443de4f7ed1bd2007a8441186a1a649ba73e47b

1.2 Issue Statistic

Item	Count	Fixed	Pending	Confirmed
Total	4		3	1
Minor	1		1	
Medium	3		2	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.
- **Confirmed:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

2 Summary of Findings

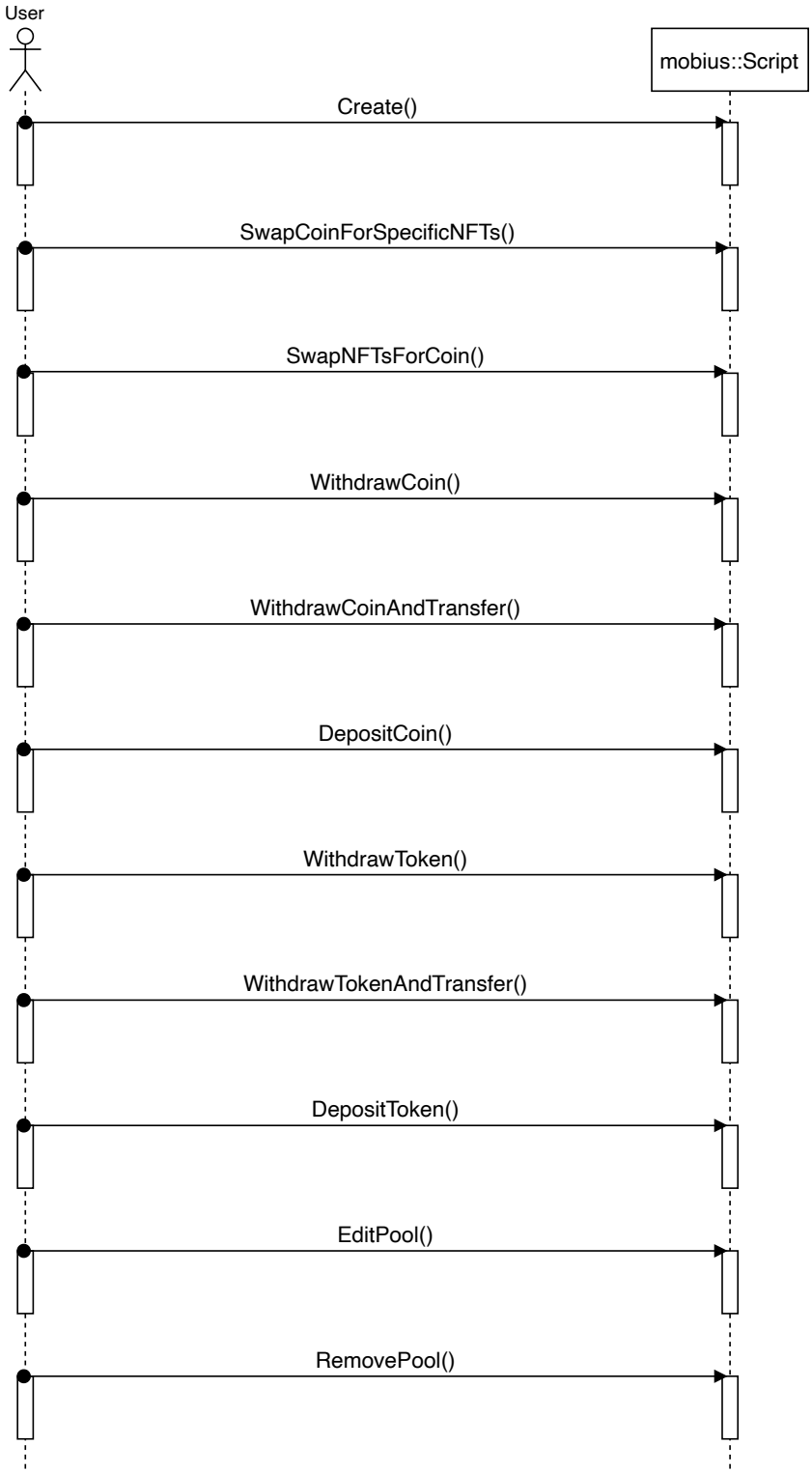
Mobius is a minimized NFT Automated Market Maker (AMM) protocol that facilitates swaps between NFTs and tokens using different pricing curves. Our team mainly focused on reviewing the Code Security and normative, then conducted code running tests and business logic security tests on the main net. During the testing process, our team also maintains close communication with the project team to ensure that we have a correct understanding of business requirements. As a result, our team found a total of 4 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 `Mobius-NFT-AMM` Smart Contract：

User

- Users can create a `Pool`.
- Users can buy and sell NFT.
- Pool creators can withdraw coins from the `Pool` and transfer them to the address.
- Pool creators can deposit coins into the `Pool`.
- Pool creators can withdraw NFT from the `Pool` and transfer them to the address.
- Pool creators can deposit NFT into the `Pool`.
- Pool creators can modify and delete `Pool`.



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 `curve` is not validated when creating and editing `Pool`

Severity: Medium

Status: Confirmed

Descriptions: In the two functions `Pool::SwapCoinForSpecificNFTs` and `Pool::SwapNFTsForCoin`, it is verified that the `curve` can only be `1` or `2`. If the `curve` is set to a value other than `1` and `2` when creating and editing the `Pool`, then the NFT in this `Pool` cannot be used. Trade until the `curve` is modified to the correct value.

Code Location: mobius-nft-amm/sources/Pool.move, line 77.

Suggestion: Add assert to verify `curve` in `Pool::create`.

Pool.move

```
const ERR_CURVE = 111;
public fun create(sender: &signer, curve: u8, creators: vector<address>, colle
ction_names: vector<String>, token_names: vector<String>, property_versions: v
ector<u64>, deposit_coin_amount: u64, spot_price: u64, delta: u64, fee: u64, t
ype: u8) acquires Pools, PoolGuid {
    assert!(curve == 1 || curve == 2, error::invalid_argument(ERR_CURVE));
    ......
}
```

Confirmed: Communicate with the development team and confirm this problem, and the development team will not modify it in the current version.

6.2 The judgment condition in the function is always not true

Severity: Medium

Status: Confirmed

Descriptions: `ROYALTY` is used as a judgment condition in `Pool::SwapCoinForSpecificNFTs` and `Pool::SwapNFTsForCoin`, but the value of `ROYALTY` is `false`, and it is a constant that will not be changed, which will cause the logic in the condition to never be executed and consume more A lot of gas.

Code Location: mobius-nft-amm/sources/Pool.move, line 133, 178.

Pool.move

```
public fun SwapCoinForSpecificNFTs(sender: &signer, target_pool_index: u128, c
reators: vector<address>, collection_names: vector<String>, token_names: vecto
r<String>, property_versions: vector<u64>) acquires Pools {
    ......
    if (ROYALTY) {
        ......
    };
    ......
}
```

Suggestion: Delete the code that uses `ROYALTY` as the judgment condition in `Pool::SwapCoinForSpecificNFTs` and `Pool::SwapNFTsForCoin`.

Confirmed: After communicating with the development team, their initial idea was to make a switch. When `ROYALTY` is set to `true`, the logic in the condition can be correctly executed, so this judgment is reserved for now.

6.3 Multiplication and cast causes overflow

Severity: Medium

Status: Confirmed

Descriptions: When the function `mul_div` is passed in for example `MAX_U64`, `MAX_U64`, 1, `r` is greater than `MAX_U64`, and it will abort at `(r as u64)`. The function `mul_div_u128` is the same, at the same time in `mul_div_u128`, let `r = x * y / z`; overflow abort will occur when `x * y` is greater than `MAX_U128`.

Code Location: mobius-nft-amm/sources/lib/Math.move, line 35, 47.

Math.move

```
/// Implements: `x` * `y` / `z`.
public fun mul_div(x: u64, y: u64, z: u64): u64 {
    assert!(z != 0, ERR_DIVIDE_BY_ZERO);
    let r = (x as u128) * (y as u128) / (z as u128);
    (r as u64)
}
......
/// Implements: `x` * `y` / `z`.
public fun mul_div_u128(x: u128, y: u128, z: u128): u64 {
    assert!(z != 0, ERR_DIVIDE_BY_ZERO);
    let r = x * y / z;
    (r as u64)
}
```

Suggestion: Add an assertion whether overflow occurs at the position where overflow may occur.

Math.move

```
const ERR_U64_OVERFLOW : u64 = 0;
const ERR_U128_OVERFLOW : u64 = 2;
/// Implements: `x` * `y` / `z`.
public fun mul_div(x: u64, y: u64, z: u64): u64 {
    assert!(z != 0, ERR_DIVIDE_BY_ZERO);
    let r = (x as u128) * (y as u128) / (z as u128);
    assert!(r <= MAX_U64, ERR_U64_OVERFLOW);
    (r as u64)
}
......
/// Implements: `x` * `y` / `z`.
public fun mul_div_u128(x: u128, y: u128, z: u128): u64 {
    assert!(z != 0, ERR_DIVIDE_BY_ZERO);
    assert!(x * y <= MAX_U128, ERR_U128_OVERFLOW);
    let r = x * y / z;
    assert!(r <= MAX_U64, ERR_U64_OVERFLOW);
    (r as u64)
}
```

Confirmed: Communicate with the development team and confirm this problem, and the development team will not modify it in the current version.

6.4 Inefficient `pow` function

Severity: Minor

Status: Confirmed

Descriptions: Using the `while` loop to calculate `pow` is inefficient. The official has implemented a more efficient `pow` function. It is recommended to use the official `pow` function.

Code Location: mobius-nft-amm/sources/lib/Math.move, line 71.

Math.move

```
/// Returns degree^n.
public fun pow(degree: u64, n: u64): u64 {
    let res = 1;
    let i = 0;
    while (
        i < n
    ) {
        res = res * degree;
        i = i + 1;
    };
    res
}
```

Suggestion: Use the function `aptos_std::math64::pow` instead.

Confirmed: Communicate with the development team and confirm this problem, and the development team will not modify it in the current version.

Appendix 1 – Files in Scope

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

Files	SHA–1 Hash
sources/Pool.move	2d4594553c6e77c3420a4cbc388e6b21118121f0
sources/Curve/LinearCurve.move	20a55fe16438ed45abfb6ef6067420743ef6bcd9
sources/Curve/ExponentialCurve.move	202bb2d25cf22a18407b52b72c7641dbbb6bf398
sources/Script.move	c049beb4e37e52a4e5873e568110ee2fafd90899
sources/lib/Math.move	cad2b2a545df7d8d1e15c7560c71b52afb353d30

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