MOVEBIT
Securing the Move Ecosystem

Mokshya/Wapal Aptos NFT Mint Smart Contract **Audit Report**

https://twitter.com/movebit_

contact@movebit.xyz

Mokshya/Wapal Aptos NFT Mint Smart Contract Audit Report

1 Executive Summary

1.1 Project Information

Description	Random NFT mint smart contract for Aptos Blockchain.
Type	NFT
Auditors	MoveBit
Timeline	Mar 3, 2023 – Mar 9, 2023
Languages	Move
Platform	Aptos
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/mokshyaprotocol/aptos-nft-random-mint
Commits	14c9e6e56041cfea3bd025fc056b3ca4cb410d43 353e25a98726f5af97fdd638f17e1a88cc16b3d0

Updates	May 12, 2023, reviewed commit 49eb92eef51d1cf5e089409ca413ac5eca9d25d4

1.2 Files in Scope

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

ID	Files	SHA-1 Hash
BTV	sources/bit_vector.move	b7131f93349b8bdcfda62bba63cbf9163dff85e1
BKT	sources/bucket_table.move	c7a814a44b2c95fe7f711e13ae8269a796add086
CDM	sources/candymachine.move	287dbdf0152b52302ef32a30c789ce0ec0d6a7d9
MKP	sources/merkle_proof.move	f471046c5245d19a97ee2960af7ae9a453c01100

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	10	10	
Informational			
Minor	5	5	
Medium	3	3	
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 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 **Mokshya** to identify any potential issues and vulnerabilities in the source code of the **Mokshya/Wapal Aptos NFT Mint** 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 **10** issues of varying severity, listed below.

ID	Title	Severity	Status
CDM-01	Wrong Condition in `assert`	Medium	Fixed
CDM-02	Unverified `public_sale_mint_time` Must be Greater Than `presale_mint_time`	Medium	Fixed
CDM-03	Inappropriate `borrow`	Minor	Fixed
CDM-04	The Verification Conditions of `assert` and `if` are Repeated	Minor	Fixed

CDM–05	Redundant Conditional Statement	Minor	Fixed
CDM–06	Potential Data Conflict in `mint_from_merkle` and `mint_script`	Major	Fixed
CDM–07	Inadequate Test Code Maintenance	Minor	Fixed
CDM–08	Missing Emit Events	Minor	Fixed
CDM–09	Record `total_apt` Error	Major	Fixed
CDM–10	Incorrect Conditional Statement.	Medium	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the `Mokshya/Wapal Aptos NFT Mint` SmartContract:

Creator

- Creators can create collections and initialize configuration parameters through `init_candy()` .
- Creators can reset `merkle_root` through `set_root()` .
- Creator can pause and resume minting through `pause_resume_mint()` .
- Creator can update the configuration of candy through `update_candy()` .

User

- User can mint NFTs through `mint_script()` .

whitelist user

- Whitelisted users can mint NFTs through `mint_from_merkle()` .

4 Findings

CDM–01 Wrong Condition in `assert`

Severity: Medium

Status: Fixed

Code Location: sources/candymachine.move, line 308.

Descriptions: In the function `update_candy` , the parameter `royalty_points_denominator` judges the wrong condition here, which will lead to never being able to update `candy_data.royalty_points_denominator` .

candymachine.move

```
public entry fun update_candy(
     account: &signer,
     candymachine: address,
     royalty_points_denominator: u64,
     royalty_points_numerator: u64,
     presale_mint_time: u64,
     public_sale_mint_price: u64,
     presale_mint_price: u64,
     public_sale_mint_time: u64,
)acquires ResourceInfo,CandyMachine{
     ...
     assert!(royalty_points_denominator == 0, EINVALID_ROYALTY_NUMERATOR_DENOMI
NATOR);
     if (royalty_points_denominator>0){
         candy_data.royalty_points_denominator = royalty_points_denominator
     };
     ...
}
```

Suggestion: The if condition below has judged `royalty_points_denominator` , directly delete this assertion `assert!(royalty_points_denominator == 0, EINVALID_ROYALTY_NUMERATOR_DENOMINATOR);` .

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

CDM–02 Unverified `public_sale_mint_time` Must be Greater Than `presale_mint_time`

Severity: Medium

Status: Fixed

Code Location: sources/candymachine.move, lines 72, 292.

Descriptions: In the function `candymachine::mint_from_merkle`, it is known that `public_sale_mint_time` must be greater than `presale_mint_time`, but this is not verified when creating and modifying `CandyMachine`. If `public_sale_mint_time` is less than or equal to `presale_mint_time` when creating `CandyMachine`, whitelist users will not be able to mint.

Suggestion: Add `assert` to the functions `candymachine::init_candy` and `candymachine::update_candy` to verify that `public_sale_mint_time` must be greater than `presale_mint_time`.

candymachine.move

```
public entry fun init_candy(
    account: &signer,
    collection_name: String,
    collection_description: String,
    baseuri: String,
    royalty_payee_address:address,
    royalty_points_denominator: u64,
    royalty_points_numerator: u64,
    presale_mint_time: u64,
    public_sale_mint_time: u64,
    presale_mint_price: u64,
    public_sale_mint_price: u64,
    total_supply:u64,
    collection_mutate_setting:vector<bool>,
    token_mutate_setting:vector<bool>,
    public_mint_limit: u64,
    merkle_root: vector<u8>,
    seeds: vector<u8>
){
    ......
    assert!(public_sale_mint_time > presale_mint_time && presale_mint_time >=
now,EINVALID_MINT_TIME);
    ......
}
```

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

CDM-03 Inappropriate `borrow`

Severity: Minor

Status: Fixed

Code Location: sources/candymachine.move, line 133, 148, sources/bucket_table.move, line 149.

Descriptions: In the function `candymachine::mint_script`, the resource `CandyMachine` is obtained through `borrow_global_mut`, but there is no need to modify `CandyMachine` in this function. Using `borrow_global_mut` may be risky, and the function `candymachine::mint_from_merkle` also has this problem. The same problem is similar to using `table_with_length::borrow_mut` in the function `candymachine::bucket_table::borrow`.

candymachine.move

```
public entry fun mint_script(
    receiver: &signer,
    candymachine: address,
)acquires ResourceInfo, CandyMachine,MintData,PublicMinters{
    let candy_data = borrow_global_mut<CandyMachine>(candymachine);
    let mint_price = candy_data.public_sale_mint_price;
    let now = aptos_framework::timestamp::now_seconds();
    assert!(now > candy_data.public_sale_mint_time, ESALE_NOT_STARTED);
    mint(receiver,candymachine,mint_price)
}
```

bucket_table.move

```
public fun borrow<K: copy + drop, V>(map: &mut BucketTable<K, V>, key: K): &V
{
    let index = bucket_index(map.level, map.num_buckets, sip_hash_from_value(&
key));
    let bucket = table_with_length::borrow_mut(&mut map.buckets, index);
    ......
}
```

Suggestion: Replace `borrow_global_mut` with `borrow_global`.

candymachine.move

```
public entry fun mint_script(
    receiver: &signer,
    candymachine: address,
)acquires ResourceInfo, CandyMachine,MintData,PublicMinters{
    let candy_data = borrow_global<CandyMachine>(candymachine);
    let mint_price = candy_data.public_sale_mint_price;
    let now = aptos_framework::timestamp::now_seconds();
    assert!(now > candy_data.public_sale_mint_time, ESALE_NOT_STARTED);
    mint(receiver,candymachine,mint_price)
}
```

Replace `table_with_length::borrow_mut` with `table_with_length::borrow`.

bucket_table.move

```
public fun borrow<K: copy + drop, V>(map: &BucketTable<K, V>, key: K): &V {
    let index = bucket_index(map.level, map.num_buckets, sip_hash_from_value(&
key));
    let bucket = table_with_length::borrow(&map.buckets, index);
    ......
}
```

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

CDM-04 The Verification Conditions of `assert` and `if` are Repeated

Severity: Minor

Status: Fixed

Code Location: sources/candymachine.move, line 159, 305.

Descriptions: In the function `candymachine::mint_from_merkle`, use `assert` to verify that `is_whitelist_mint` is true to continue to execute the code, and then repeat the judgment through `if`.

In the function `candymachine::update_candy`, `assert` has been used to verify that `public_sale_mint_time >= now && presale_mint_time >= now`, and the judgment is repeated through `if` below.

candymachine.move

```
public entry fun mint_from_merkle(
    receiver: &signer,
    candymachine: address,
    proof: vector<vector<u8>>,
    mint_limit: u64
) acquires ResourceInfo,MintData,PublicMinters,CandyMachine,Whitelist{
    ......
    assert!(is_whitelist_mint, WhitelistMintNotEnabled);
    if(is_whitelist_mint){
        // No need to check limit if mint limit = 0, this means the minter ca
n mint unlimited amount of tokens
        if(mint_limit != 0){
            let whitelist_data = borrow_global_mut<Whitelist>(candymachine);
            if (!bucket_table::contains(&whitelist_data.minters, &receiver_add
r)) {
                // First time minting mint limit = 0
                bucket_table::add(&mut whitelist_data.minters, receiver_addr,
0);
            };
            let minted_nft = bucket_table::borrow_mut(&mut whitelist_data.mint
ers, receiver_addr);
            assert!(*minted_nft != mint_limit, MINT_LIMIT_EXCEED);
            *minted_nft = *minted_nft + 1;
            mint_data.total_apt=candy_data.presale_mint_price;
            mint_price = candy_data.presale_mint_price
        };
        mint(receiver,candymachine,mint_price)
    };
}

public entry fun update_candy(
    account: &signer,
    candymachine: address,
    royalty_points_denominator: u64,
    royalty_points_numerator: u64,
    presale_mint_time: u64,
    public_sale_mint_price: u64,
    presale_mint_price: u64,
    public_sale_mint_time: u64,
)acquires ResourceInfo,CandyMachine{
    let account_addr = signer::address_of(account);
    let resource_data = borrow_global<ResourceInfo>(candymachine);
    let now = aptos_framework::timestamp::now_seconds();
    assert!(public_sale_mint_time >=  now && presale_mint_time >= now, EINVALI
D_MINT_TIME);
```

```
        ...
        if (presale_mint_time>0){
            candy_data.presale_mint_time = presale_mint_time
        };
        if (public_sale_mint_time>0){
            candy_data.public_sale_mint_time = public_sale_mint_time
        };
        ...
    }
```

Suggestion: Remove the `if` judgment.

candymachine.move

```
public entry fun mint_from_merkle(
    receiver: &signer,
    candymachine: address,
    proof: vector<vector<u8>>,
    mint_limit: u64
) acquires ResourceInfo,MintData,PublicMinters,CandyMachine,Whitelist{
    ......
    assert!(is_whitelist_mint, WhitelistMintNotEnabled);
    // No need to check limit if mint limit = 0, this means the minter can min
t unlimited amount of tokens
    if(mint_limit != 0){
        let whitelist_data = borrow_global_mut<Whitelist>(candymachine);
        if (!bucket_table::contains(&whitelist_data.minters, &receiver_addr))
{
            // First time minting mint limit = 0
            bucket_table::add(&mut whitelist_data.minters, receiver_addr, 0);
        };
        let minted_nft = bucket_table::borrow_mut(&mut whitelist_data.minters,
 receiver_addr);
        assert!(*minted_nft != mint_limit, MINT_LIMIT_EXCEED);
        *minted_nft = *minted_nft + 1;
        mint_data.total_apt=candy_data.presale_mint_price;
        mint_price = candy_data.presale_mint_price
    };
    mint(receiver,candymachine,mint_price)
}

public entry fun update_candy(
    account: &signer,
    candymachine: address,
    royalty_points_denominator: u64,
    royalty_points_numerator: u64,
    presale_mint_time: u64,
    public_sale_mint_price: u64,
    presale_mint_price: u64,
    public_sale_mint_time: u64,
)acquires ResourceInfo,CandyMachine{
    let account_addr = signer::address_of(account);
    let resource_data = borrow_global<ResourceInfo>(candymachine);
    let now = aptos_framework::timestamp::now_seconds();
    assert!(public_sale_mint_time >=  now && presale_mint_time >= now, EINVALI
D_MINT_TIME);
    ...
    candy_data.presale_mint_time = presale_mint_time;
```

```
        candy_data.public_sale_mint_time = public_sale_mint_time;
        ...
    }
```

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

CDM–05 Redundant Conditional Statement

Severity: Minor

Status: Fixed

Code Location: sources/candymachine.move, lines 398–402.

Descriptions: Whether to enter the `if (nfts < 1024)` statement in the function `candymachine::create_bit_mask` has no effect on the values of `full_buckets` and `remaining`.

candymachine.move

```
    fun create_bit_mask(nfts: u64): vector<BitVector>
    {
        let full_buckets = nfts/1024;
        let remaining =nfts-full_buckets*1024;
        if (nfts < 1024)
        {
            full_buckets=0;
            remaining= nfts;
        };
        ......
    }
```

Suggestion: Delete the `if (nfts < 1024)` statement.

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

CDM–06 Potential Data Conflict in `mint_from_merkle` and `mint_script`

Severity: Major

Status: Fixed

Code Location: sources/candymachine.move, line 186.

Descriptions: When initializing the candy, if the values of `presale_mint_price` and `public_sale_mint_price` are set equal, then whitelisted users calling the `mint_from_merkle` function to mint NFTs will trigger the `initialize_and_create_public_minter` function, which will result in the reduction of their public sale opportunities. We would like to know if it aligns with the design.

candymachine.move

```
fun mint(
    receiver: &signer,
    candymachine: address,
    mint_price: u64
)acquires ResourceInfo, CandyMachine,PublicMinters,MintData{
    ...
    if(candy_data.public_sale_mint_price == mint_price && candy_data.public_mi
nt_limit != 0){
        initialize_and_create_public_minter(&resource_signer_from_cap,candy_da
ta,receiver_addr,candymachine);
        mint_data.total_apt=candy_data.public_sale_mint_price;
    };
    ...
}
```

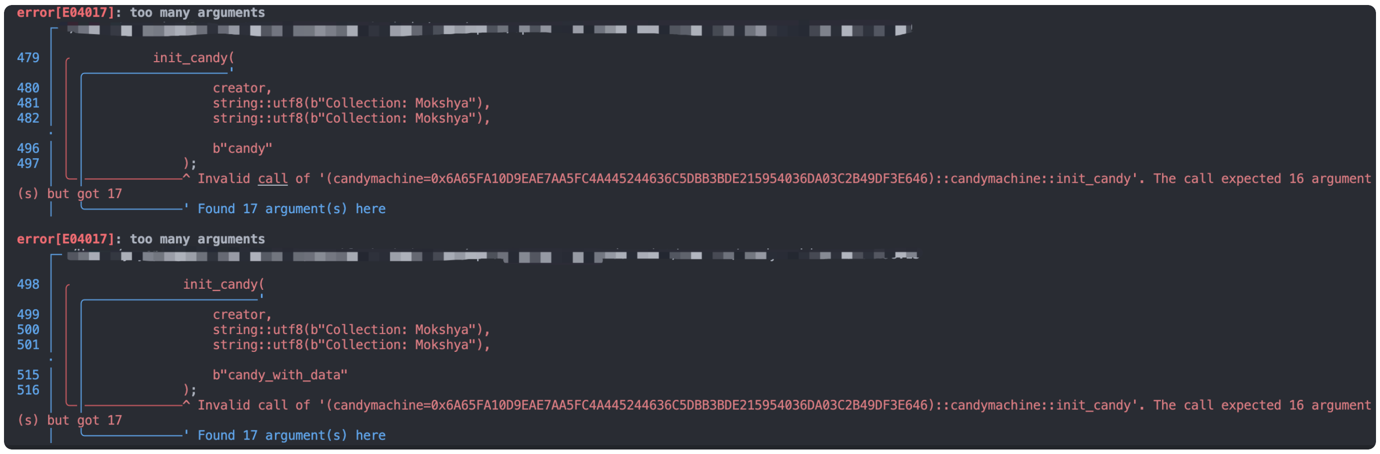
Resolution: The client has followed our suggestion and fixed the issue.

CDM–07 Inadequate Test Code Maintenance

Severity: Minor

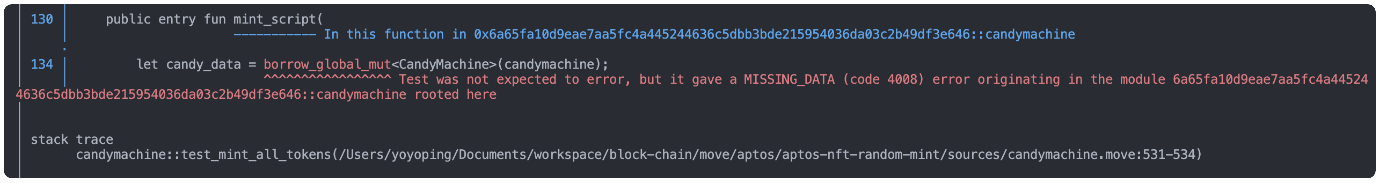
Status: Fixed

Descriptions: When running test cases, an error occurred with the following message:

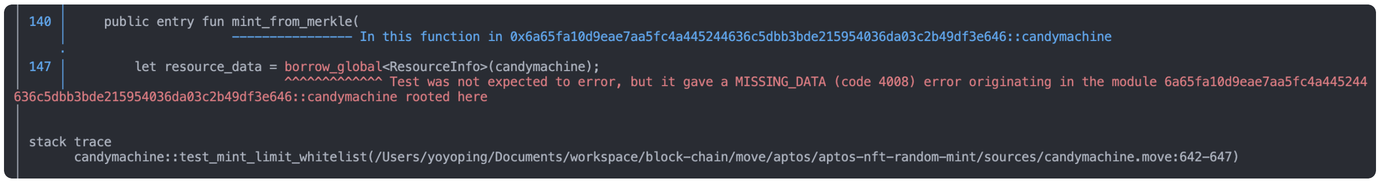


```
error[E04017]: too many arguments
    ┌─ [redacted]
    │
479 │           init_candy(
    │
480 │                   creator,
481 │                   string::utf8(b"Collection: Mokshya"),
482 │                   string::utf8(b"Collection: Mokshya"),
    ·
496 │                   b"candy"
497 │               );
    │               ^ Invalid call of '(candymachine=0x6A65FA10D9EAE7AA5FC4A445244636C5DBB3BDE215954036DA03C2B49DF3E646)::candymachine::init_candy'. The call expected 16 argument
(s) but got 17
    │               ' Found 17 argument(s) here

error[E04017]: too many arguments
    ┌─ [redacted]
    │
498 │               init_candy(
    │
499 │                   creator,
500 │                   string::utf8(b"Collection: Mokshya"),
501 │                   string::utf8(b"Collection: Mokshya"),
    ·
515 │                   b"candy_with_data"
516 │               );
    │               ^ Invalid call of '(candymachine=0x6A65FA10D9EAE7AA5FC4A445244636C5DBB3BDE215954036DA03C2B49DF3E646)::candymachine::init_candy'. The call expected 16 argument
(s) but got 17
    │               ' Found 17 argument(s) here
```

The above error message clearly indicates that there are extra parameters. After fixing this issue, a new error message appeared:



```
130 │     public entry fun mint_script(
    │                      ----------- In this function in 0x6a65fa10d9eae7aa5fc4a445244636c5dbb3bde215954036da03c2b49df3e646::candymachine
    ·
134 │         let candy_data = borrow_global_mut<CandyMachine>(candymachine);
    │                          ^^^^^^^^^^^^^^^^^ Test was not expected to error, but it gave a MISSING_DATA (code 4008) error originating in the module 6a65fa10d9eae7aa5fc4a44524
4636c5dbb3bde215954036da03c2b49df3e646::candymachine rooted here


stack trace
      candymachine::test_mint_all_tokens(/Users/yoyoping/Documents/workspace/block-chain/move/aptos/aptos-nft-random-mint/sources/candymachine.move:531-534)
```



```
140 │     public entry fun mint_from_merkle(
    │                      ---------------- In this function in 0x6a65fa10d9eae7aa5fc4a445244636c5dbb3bde215954036da03c2b49df3e646::candymachine
    ·
147 │         let resource_data = borrow_global<ResourceInfo>(candymachine);
    │                             ^^^^^^^^^^^^^ Test was not expected to error, but it gave a MISSING_DATA (code 4008) error originating in the module 6a65fa10d9eae7aa5fc4a445244
636c5dbb3bde215954036da03c2b49df3e646::candymachine rooted here


stack trace
      candymachine::test_mint_limit_whitelist(/Users/yoyoping/Documents/workspace/block-chain/move/aptos/aptos-nft-random-mint/sources/candymachine.move:642-647)
```

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

CDM-08 Missing Emit Events

Severity: Minor

Status: Fixed

Code Location: sources/candymachine.move, line 278.

Descriptions: When users update the candy or token, there is a lack of event, which means that changes cannot be monitored from the outside and there is no way to notify the outside world in real-time that it has been updated.

Suggestion: Add events after the update is complete to enable external monitoring.

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

CDM-09 Record `total_apt` Error

Severity: Major

Status: Fixed

Code Location: sources/candymachine.move, line 178, 195.

Description: According to the code logic, the variable `total_apt` is used to record the price of the last minted NFT, but the name suggests that it should be used to record the total amount of minted NFTs.

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

CDM-10 Incorrect Conditional Statement.

Severity: Medium

Status: Fixed

Code Location: sources/candymachine.move, line 315.

Descriptions: In the function `update_candy()`, there is an error in the conditional statement that updates `presale_mint_price`.

candymachine.move

```
if (public_sale_mint_price>0){
    candy_data.presale_mint_price = presale_mint_price
};
```

Suggestion: Modify the conditional statement.

candymachine.move

```
if (presale_mint_price>0){
    candy_data.presale_mint_price = presale_mint_price
};
```

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

MOVEBIT

Securing the Move Ecosystem

https://twitter.com/movebit_

contact@movebit.xyz