MOVEBIT
Securing the Move Ecosystem

Starcoin Framework **Audit Report**

https://twitter.com/movebit_

contact@movebit.xyz

Starcoin Framework Audit Report

1 Executive Summary

1.1 Project Information

Type	Framework
Auditors	MoveBit
Timeline	2022-08-29 to 2022-09-21
Languages	Move
Methods	Architecture Review, Unit Testing, Formal Verification, Manual Review
Specification	SIP <https://github.com/starcoinorg/sips>
Source Code	V11 <https://github.com/starcoinorg/starcoin-framework/tree/v11>

1.2 Issue Statistic

Item	Count	Fixed	Pending
Total	21		21
Minor	16		16
Medium	4	1	3
Major	1	1	
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

Starcoin-framework serves as a standard library of the public chain Starcoin. This is a large library consisting of 69 Move source files and more than 70 modules (e.g., Account, Token, STC, Config, DAO, NFT, Oracle, Genesis, and Block). Prior to this audit work, we read the Starcoin SIPs and other developer resources in advance. We first took a review of the framework architecture, then mainly focused on the code review and formal verification with the Move Prover. We've been in close contact with the Starcoin team for the past few weeks. As a result, we found a total of 21 issues. We had an extensive discussion of all the issues during a Zoom meeting with the Starcoin team. Some of the problems were already fixed in later commits, the others are in plan to be addressed soon.

We added formal specifications for most of the functions, except for native functions and some functions that contain special elements that can't be reasoned about (e.g., runtime type information, bitwise operators). All the verification code will be submitted as PR to the code repository, and eventually get merged by the Starcoin team in later revisions.

Here's the list of general suggestions:

- Many files contain redundant code that is meant to optimize the program, but is actually suboptimal and more computationally expensive than a simpler implementation, resulting in more gas usage;
- Coding style is not consistent within the project, including line width limit, code indentation;

- Some method names do not conform to usual English grammar.

3 MoveBit Audit BreakDown

MoveBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow
- 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

4 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 in the conventions in the "Audit Objective", and that can expand to the context 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 see **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, and they should actively cooperate (which may include 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 time.

5 Findings

5.1 Incorrect while statements without loop counter updated

Severity: Major

Status: Fixed

Descriptions: There are two while statements without loop counter correctly updated. It produces an infinite loop. If it gets executed, it will run out of gas for the transaction.

Code Location: sources/signature.move, line 53,62

```
public fun new(bytes: vector<u8>): EVMAddress{
    let len = Vector::length(&bytes);
    let bytes = if (len > EVM_ADDR_LENGTH){
        let new_bytes = Vector::empty<u8>();
        let i = 0;
        while (i < EVM_ADDR_LENGTH) {
            Vector::push_back(&mut new_bytes, *Vector::borrow(&bytes, i));
        };
        new_bytes
    }else if (len == EVM_ADDR_LENGTH){
        bytes
    }else{
        let i = 0;
        let new_bytes = Vector::empty<u8>();
        while (i < EVM_ADDR_LENGTH - len) {
            // pad zero to address
            Vector::push_back(&mut new_bytes, 0);
        };
        Vector::append(&mut new_bytes, bytes);
        new_bytes
    };
    EVMAddress{
        bytes
    }
}
```

Suggestion: Add statements to change the loop variable `i`, and make sure the loop terminates at right condtion.

5.2 Inconsistent sign flag for `SignedInteger64` for zero value

Severity: Medium
Status: Pending
Descriptions: If the return value of `add_u64` and `sub_u64` is 0, it may have a different sign flag. For example:

```
public fun sub_u64(num: u64, minus: SignedInteger64): SignedInteger64
sub_u64(0, -0) returns +0
sub_u64(0, +0) returns -0

public fun add_u64(num: u64, addend: SignedInteger64): SignedInteger64
add_u64(1, -1), returns -0
add_u64(0, +0), returns +0
add_u64(0, -0), returns -0
```

Code Location: sources/SignedInteger64.move, line 29,44

```
/// Sub: `num - minus`
public fun sub_u64(num: u64, minus: SignedInteger64): SignedInteger64 {
    if (minus.is_negative) {
        let result = num + minus.value;
        SignedInteger64 { value: (result as u64), is_negative: false }
    } else {
        if (num > minus.value)  {
            let result = num - minus.value;
            SignedInteger64 { value: (result as u64), is_negative: false }
        }else {
            let result = minus.value - num;
            SignedInteger64 { value: (result as u64), is_negative: true }
        }
    }
}
/// Add: `num + addend`
public fun add_u64(num: u64, addend: SignedInteger64): SignedInteger64 {
    if (addend.is_negative) {
       if (num > addend.value)  {
           let result = num - addend.value;
           SignedInteger64 { value: (result as u64), is_negative: false }
       }else {
           let result = addend.value - num;
           SignedInteger64 { value: (result as u64), is_negative: true }
       }
    } else {
         let result = num + addend.value;
         SignedInteger64 { value: (result as u64), is_negative: false }
    }
}
```

Suggestion: Change the code logic, and make sure it has the same sign flag when the return value is 0.

5.3 Unchecked range of carry bit in function `adc`

Severity: Medium

Status: Pending

Descriptions: The carry bit of two u64 numbers should logically be 0 or 1. The argument `carry` for public function `adc()` may be invalid, and it may cause a wrong result.

Code Location: sources/U256.move, line 16

```
/// a + b, with carry
public fun adc(a: u64, b: u64, carry: &mut u64) : u64 {
    let (a1, a0) = split_u64(a);
    let (b1, b0) = split_u64(b);
    let (c, r0) = split_u64(a0 + b0 + *carry);
    let (c, r1) = split_u64(a1 + b1 + c);
    *carry = c;
    combine_u64(r1, r0)
}
```

Suggestion: Add an assertion for argument `carry` , and make sure it is valid (0 or 1).

5.4 Unchecked range of borrow bit in function `sbb`

Severity: Medium

Status: Fixed

Descriptions: The borrow bit of two u64 numbers should logically be 0 or 1. The argument `borrow` for public function `sbb()` may be invalid, and it may cause a wrong result.

Code Location: sources/U256.move, line 25

```
/// a - b, with borrow
public fun sbb(a: u64, b: u64, borrow: &mut u64): u64 {
    let (a1, a0) = split_u64(a);
    let (b1, b0) = split_u64(b);
    let (b, r0) = split_u64((1 << 32) + a0 - b0 - *borrow);
    let borrowed = if(b==0) {1} else {0};
    let (b, r1) = split_u64((1 << 32) + a1 - b1 - borrowed);
    *borrow = if(b==0) {1} else {0};

    combine_u64(r1, r0)
}
```

Suggestion: Add assertion for argument borrow, make sure it is valid (0 or 1).

5.5 Existence of public module functions that are unusable for general accounts

Severity: Medium

Status: Pending

Descriptions: Some `block` and `transaction` related module functions must be called with account `GENESIS_ADDRES` . Since these functions are not allowed for normal users to access, remove or hide them away from normal users. Removing or hiding these functions can minimize the potential risk.

Code Location: sources/TransactionManager.move, sources/Block.move, sources/BlockReward.move, sources/Account.move

```
// below functions can't be called from normal users

TransactionManager::block_prologue, TransactionManager::epilogue
Block::process_block_metadata
BlockReward::process_block_reward
Account::txn_prologue, Account::txn_epilogue
```

Suggestion: Remove or Hide the unusable module for STD library users.

5.6 Existence of unused private functions in module `U256`

Severity: Minor
Status: Pending
Descriptions: There are unused private functions in module U256, these functions are `add_nocarry` and `sub_noborrow` .

Code Location: sources/U256.move, line 256,272

```
/// move implementation of native_add.
fun add_nocarry(a: &mut U256, b: &U256) {
    let carry = 0;
    let idx = 0;
    let len = (WORD as u64);
    while (idx < len) {
        let a_bit = Vector::borrow_mut(&mut a.bits, idx);
        let b_bit = Vector::borrow(&b.bits, idx);
        *a_bit = StarcoinFramework::Arith::adc(*a_bit, *b_bit, &mut carry);
        idx = idx + 1;
    };

    // check overflow
    assert!(carry == 0, 100);
}

/// move implementation of native_sub.
fun sub_noborrow(a: &mut U256, b: &U256) {
    let borrow = 0;
    let idx = 0;
    let len =(WORD as u64);
    while (idx < len) {
        let a_bit = Vector::borrow_mut(&mut a.bits, idx);
        let b_bit = Vector::borrow(&b.bits, idx);
        *a_bit = StarcoinFramework::Arith::sbb(*a_bit, *b_bit, &mut borrow);
        idx = idx + 1;
    };

    // check overflow
    assert!(borrow == 0, 100);

}
```

Suggestion: Remove these functions.

5.7 Too many TODO labels

Severity: Minor

Status: Pending

Descriptions: There are too many TODO labels in the project. We inspected them case by case, some are already done, and some need to be addressed in the future.

Code Location: sources/U256.move, sources/Genesis.move, sources/Offer.move, and many other files.

Suggestion: The dev team needs to search the TODO in the code, review all the labels, remove the items that are actually completed, and work on the left ones.

5.8 Duplicated functions with identical functionality `is_accepts_token` and `is_accept_token` in moudle `Account`

Severity: Minor

Status: Pending

Descriptions: These two functions confuse the user. Is it one kept for compatibility?

Code Location: sources/Account.move, line 703

```
/// This is a alias of is_accept_token
public fun is_accepts_token<TokenType: store>(addr: address): bool acquires AutoAcceptToken {
    Self::is_accept_token<TokenType>(addr)
}

spec is_accepts_token {
    aborts_if false;
}

/// Return whether the account at `addr` accept `Token` type tokens
public fun is_accept_token<TokenType: store>(addr: address): bool acquires AutoAcceptToken {
    if (can_auto_accept_token(addr)) {
        true
    } else {
        exists<Balance<TokenType>>(addr)
    }
}
```

Suggestion: Add comments to address the difference between the two functions.

5.9 Chain type defined only by chain ID

Severity: Minor

Status: Pending

Descriptions: In module ChainId, determining whether the chain is dev, test, halley, proxima, barnard, and main type is not enough. The custom chain may be configured from a dev template with the main id. The wrong logic may cause an incorrect impact to VMConfig for calculating maximum_number_of_gas_units.

Code Location: sources/ChainId.move, line 45

```
// In ChainId.move
public fun is_dev(): bool acquires ChainId {
    get() == DEV_CHAIN_ID
}
public fun is_test(): bool acquires ChainId {
    get() == TEST_CHAIN_ID
}
public fun is_halley(): bool acquires ChainId {
    get() == HALLEY_CHAIN_ID
}
public fun is_proxima(): bool acquires ChainId {
    get() == PROXIMA_CHAIN_ID
}
public fun is_barnard(): bool acquires ChainId {
    get() == BARNARD_CHAIN_ID
}
public fun is_main(): bool acquires ChainId {
    get() == MAIN_CHAIN_ID
}

// In VMConfig.move
public fun gas_constants(): GasConstants {
    let min_price_per_gas_unit: u64 = if (ChainId::is_test()) { 0 }  else { 1 };
    let maximum_number_of_gas_units: u64 = 40000000;//must less than base_block_gas_lim
it

    if (ChainId::is_test() || ChainId::is_dev() || ChainId::is_halley()) {
        maximum_number_of_gas_units = maximum_number_of_gas_units * 10
    };
    GasConstants {
        global_memory_per_byte_cost: 4,
        global_memory_per_byte_write_cost: 9,
        min_transaction_gas_units: 600,
        large_transaction_cutoff: 600,
        instrinsic_gas_per_byte: 8,
        maximum_number_of_gas_units,
        min_price_per_gas_unit,
        max_price_per_gas_unit: 10000,
        max_transaction_size_in_bytes: 1024 * 128,
        gas_unit_scaling_factor: 1,
        default_account_size: 800,
    }
}
```

Suggestion: Add more condition checks for these functions, and make sure the chain type is correct.

5.10 Use of deprecated functions in module `MerkleNFTDistributor`

Severity: Minor

Status: Pending

Descriptions: In function `MerkleNFTDistributor::register` , `NFT::register` was called. In function `MerkleNFTDistributor::mint_with_cap,` `NFT::mint_with_cap` was called. There are two new version functions(`register_v2` , `mint_with_cap_v2`) in NFT, should call the new ones?

Code Location: sources/MerkleNFT.move, line 65,80

```
public fun register<NFTMeta: copy + store + drop, Info: copy + store + drop>(signer: &s
igner, merkle_root: vector<u8>, leafs: u64, info: Info, meta: Metadata): MintCapability
<NFTMeta> {
    let bitmap_count = leafs / 128;
    if (bitmap_count * 128 < leafs) {
        bitmap_count = bitmap_count + 1;
    };
    let claimed_bitmap = Vector::empty();
    let j = 0;
    while (j < bitmap_count) {
        Vector::push_back( &mut claimed_bitmap, 0u128);
        j = j + 1;
    };
    let distribution = MerkleNFTDistribution<NFTMeta>{
        merkle_root,
        claimed_bitmap
    };
    NFT::register<NFTMeta, Info>(signer, info, meta);
    move_to(signer, distribution);
    NFT::remove_mint_capability<NFTMeta>(signer)
}

public fun mint_with_cap<NFTMeta: copy + store + drop, NFTBody: store, Info: copy + sto
re + drop>(sender: &signer, cap:&mut MintCapability<NFTMeta>, creator: address, index:
u64, base_meta: Metadata, type_meta: NFTMeta, body: NFTBody, merkle_proof:vector<vector
<u8>>): NFT<NFTMeta, NFTBody>
    acquires MerkleNFTDistribution {
        let addr = Signer::address_of(sender);
        let distribution = borrow_global_mut<MerkleNFTDistribution<NFTMeta>>(creator);
        let minted = is_minted_<NFTMeta>(distribution, index);
        assert!(!minted, Errors::custom(ALREADY_MINTED));
        let leaf_data = encode_leaf(&index, &addr);
        let verified = MerkleProof::verify(&merkle_proof, &distribution.merkle_root, Ha
sh::sha3_256(leaf_data));
        assert!(verified, Errors::custom(INVALID_PROOF));
        set_minted_(distribution, index);
        let nft = NFT::mint_with_cap<NFTMeta, NFTBody, Info>(creator, cap, base_meta, t
ype_meta, body);
        return nft
    }
```

Suggestion: Take a review and decide whether should call the new ones, if not, add comments to explain the reasons.

5.11 Magic number literal

Severity: Minor
Status: Pending
Descriptions: U64 literal constant 18446744073709551615u64 is used in Bitwise.move. The decimal format is error-prone and difficult to recognize.

Code Location: sources/Bitwise.move, line 26

```
/// bit not: !x
public fun not(x: u64): u64 {
   (x ^ 18446744073709551615u64 as u64)
}
```

Suggestion: Suggest to change it to hex format 0xFFFFFFFFFFFFFFFF.

5.12 Missing documentation

Severity: Minor
Status: Pending
Descriptions: There are some design articles for some parts, eg. Account, Token, DAO, and NFT. That's not enough for an STD library. It needs more detailed documents for library users. Right now, users can only read the source code to help them use the library.

Suggestion: Add detailed documents for all modules.

5.13 Redundant loop variables in `Compare`

Severity: Minor
Status: Pending
Descriptions: In function `Compare::cmp_bytes()`, the loop variables `i1` and `i2` are redundant.

Code Location: sources/Compare.move, line 57

```
public fun cmp_bytes(v1: &vector<u8>, v2: &vector<u8>): u8 {
    let l1 = Vector::length(v1);
    let l2 = Vector::length(v2);
    let len_cmp = cmp_u64(l1, l2);
    let i1 = 0;
    let i2 = 0;
    while (i1 < l1 && i2 < l2) {
        let elem_cmp = cmp_u8(*Vector::borrow(v1, i1), *Vector::borrow(v2, i2));
        if (elem_cmp != 0) {
            return elem_cmp
        };
        // else, compare next element
        i1 = i1 + 1;
        i2 = i2 + 1;
    };
    // all compared elements equal; use length comparison to break the tie
    len_cmp
}
```

Suggestion: Remove i2, and change `*Vector::borrow(v2, i2)` to `*Vector::borrow(v2, i1)`. Also the comparison `(elem_cmp != 0)` can be changed to `(elem_cmp != EQUAL)` for clarity.

5.14 `cmp_bytes` and `cmp_bcs_bytes` function consuming too much gas when comparing large vector

Severity: Minor

Status: Pending

Descriptions: `cmp_bytes` and `cmp_bcs_bytes` calls the `cmp_u8` function for each byte, the CALL operation consumes much gas. We did a test, putting the compare codes into `cmp_bytes` and `cmp_bcs_bytes`, we got a 20% gas reduction when comparing 100 bytes once.

Code Location: sources/Compare.move, line 38,57

```
public fun cmp_bcs_bytes(v1: &vector<u8>, v2: &vector<u8>): u8 {
    let i1 = Vector::length(v1);
    let i2 = Vector::length(v2);
    let len_cmp = cmp_u64(i1, i2);

    // BCS uses little endian encoding for all integer types, so we choose to compare f
rom left
    // to right. Going right to left would make the behavior of Compare.cmp diverge fro
m the
    // bytecode operators < and > on integer values (which would be confusing).
    while (i1 > 0 && i2 > 0) {
        i1 = i1 - 1;
        i2 = i2 - 1;
        let elem_cmp = cmp_u8(*Vector::borrow(v1, i1), *Vector::borrow(v2, i2));
        if (elem_cmp != 0) return elem_cmp
        // else, compare next element
    };
    // all compared elements equal; use length comparison to break the tie
    len_cmp
}

public fun cmp_bytes(v1: &vector<u8>, v2: &vector<u8>): u8 {
    let l1 = Vector::length(v1);
    let l2 = Vector::length(v2);
    let len_cmp = cmp_u64(l1, l2);
    let i1 = 0;
    let i2 = 0;
    while (i1 < l1 && i2 < l2) {
        let elem_cmp = cmp_u8(*Vector::borrow(v1, i1), *Vector::borrow(v2, i2));
        if (elem_cmp != 0) {
            return elem_cmp
        };
        // else, compare next element
        i1 = i1 + 1;
        i2 = i2 + 1;
    };
    // all compared elements equal; use length comparison to break the tie
    len_cmp
}
```

Suggestion: Remove the `cmp_u8` call, and write the comparing codes in line.

5.15 Unnecessary type conversions

Severity: Minor

Status: Pending

Descriptions: Conversions from u64 to u64 are found in both files SignedInteger64.move and Bitwise.move. It's unnecessary and it consumes gas.

Code Location: sources/SignedInteger64.move, line 19,25,32; sources/Bitwise.move

```
public fun multiply_u64(num: u64, multiplier: SignedInteger64): SignedInteger64 {
    let product = multiplier.value * num;
    SignedInteger64 { value: (product as u64), is_negative: multiplier.is_negative }
}

/// Divide a u64 integer by a signed integer number.
public fun divide_u64(num: u64, divisor: SignedInteger64): SignedInteger64 {
    let quotient = num / divisor.value;
    SignedInteger64 { value: (quotient as u64), is_negative: divisor.is_negative }
}

/// Sub: `num - minus`
public fun sub_u64(num: u64, minus: SignedInteger64): SignedInteger64 {
    if (minus.is_negative) {
        let result = num + minus.value;
        SignedInteger64 { value: (result as u64), is_negative: false }
    } else {
        if (num > minus.value)  {
            let result = num - minus.value;
            SignedInteger64 { value: (result as u64), is_negative: false }
        }else {
            let result = minus.value - num;
            SignedInteger64 { value: (result as u64), is_negative: true }
        }
    }
}
```

Suggestion: Remove unnecessary conversions in both files SignedInteger64.move and Bitwise.move.

5.16 Unnecessary comparison in `YieldFarming`

Severity: Minor

Status: Pending

Descriptions: In function `YieldFarming::mul_u128` and `YieldFarming::div_u12`, the comparison prior to the multiply and divide operation is unnecessary, it has no effect and consumes more gas.

Code Location: sources/YieldFarming.move, line 71,82

```
fun mul_u128(a: u128, b: u128): u128 {
    if (a == 0 || b == 0) {
        return 0
    };

    a * b
}

fun div_u128(a: u128, b: u128): u128 {
    if ( b == 0) {
        abort Errors::invalid_argument(ERR_EXP_DIVIDE_BY_ZERO)
    };
    if (a == 0) {
        return 0
    };
    a / b
}
```

Suggestion: Remove the comparison `if (a == 0 || b == 0)` and `if (a == 0)` statements.

5.17 Unnecessary check in `Collection2`

Severity: Minor

Status: Pending

Descriptions: In function `Collection2::accept`, the check before `cs.anyone_can_put = true;` is unnecessary, and it consumes more gas.

Code Location: sources/Collection2.move, line 119

```
public fun accept<T: store>(signer: &signer) acquires CollectionStore {
     let addr = Signer::address_of(signer);
    if (!exists<CollectionStore<T>>(addr)){
        Self::create_collection<T>(signer, true, false);
    };
    let cs = borrow_global_mut<CollectionStore<T>>(addr);
    if (!cs.anyone_can_put) {
        cs.anyone_can_put = true;
    }
}
```

Suggestion: Remove the `if (!cs.anyone_can_put)` statement.

5.18 Public access of functions designed to be called by VM in `VMConfig`

Severity: Minor
Status: Pending
Descriptions: The functions(`instruction_schedule` , `native_schedule` , `gas_constants`) in module VMConfig seems to be called by VM, but it has public access, is it necessary?

Code Location: sources/VMConfig.move, line 74,224,297

```
public fun instruction_schedule(): vector<GasCost> {}
public fun native_schedule(): vector<GasCost> {}
public fun gas_constants(): GasConstants {}
```

Suggestion: Limit the access right for normal users, suggest removing public access rights, or remove these functions from the STD library.

5.19 Unused private functions in module `YieldFarmingV2`

Severity: Minor
Status: Pending
Descriptions: There are unused private functions in module YieldFarmingV2, these functions are `sub_u128` and `add_exp` .

Code Location: sources/YieldFarmingV2.move, line 59,78

```
fun add_exp(a: Exp, b: Exp): Exp {
    Exp {
        mantissa: add_u128(a.mantissa, b.mantissa)
    }
}
fun sub_u128(a: u128, b: u128): u128 {
    a - b
}
```

Suggestion: Remove these functions.

5.20 Wrongly named variable in module `DummyToken`

Severity: Minor
Status: Pending
Descriptions: Duplicated variable name `burn_cap` for different types(`Token::BurnCapability` and `Token::MintCapability`) in module DummyToken. This might be a copy-and-paste error.

Code Location: sources/DummyToken.move, line 32,35

```
public fun initialize(account: &signer) {
    Token::register_token<DummyToken>(
        account,
        PRECISION,
    );

    let burn_cap = Token::remove_burn_capability<DummyToken>(account);
    move_to(account, SharedBurnCapability{cap: burn_cap});

    let burn_cap = Token::remove_mint_capability<DummyToken>(account);
    move_to(account, SharedMintCapability{cap: burn_cap});
}
```

Suggestion: Change the second `burn_cap` to `mint_cap` .

5.21 Missing empty value checking for `name` in `NFT::new_meta`

Severity: Minor

Status: Pending

Descriptions: In functions(`NFT::new_meta_with_image` and `NFT::new_meta_with_image_data`), there have a check for empty name value(`assert!(!Vector::is_empty(&name), Errors::invalid_argument(ERR_CANOT_EMPTY));`), but none in `NFT::new_meta` .

Code Location: sources/NFT.move, line 160

```
public fun new_meta(name: vector<u8>, description: vector<u8>): Metadata {
    Metadata {
        name,
        image: Vector::empty(),
        image_data: Vector::empty(),
        description,
    }
}

public fun new_meta_with_image(name: vector<u8>, image: vector<u8>, description: vector
<u8>): Metadata {
    assert!(!Vector::is_empty(&name), Errors::invalid_argument(ERR_CANOT_EMPTY));
    assert!(!Vector::is_empty(&image), Errors::invalid_argument(ERR_CANOT_EMPTY));
    Metadata {
        name,
        image,
        image_data: Vector::empty(),
        description,
    }
}
```

Suggestion: Add the same assertion check in function `NFT::new_meta` .

6 Prover Formal Verification

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

Account

General Descriptions

Every account has the resource `Account` .

```
1  some address
2  |- Account
3  |  |- auth_key
4  |  |- withdrawal_capability
5  |  `- key_rotation_capability
6  |- Balance<T>
7  |  `- Token<T>
8  |- SignerDelegated
9  `- AutoAcceptToken
```

The important struct of this module is shown above in the tree diagram. Inside `Account` (some resources/fields are ignored because they're not used in verification anywhere), there're two optional capabilities: `withdrawal_capability` and `key_rotation_capability`, each of which possibly contains an address. That is, at most one address can withdraw from the associated account. The same rule applies for changing authentication key. This modules controls account creation and other things such as balance withdraw/deposit/transfer.

Formally Verified Properties

- One can only withdraw an amount of tokens within the limit of account balance
- Withdrawal actions can only be performed with proper capability
- `SignerDelegated` and `SignerCapability` can only be stored under addresses, i.e., cannot be member of any other data type (ensured by the `key` ability)

Propertiy can be modeled yet not verified

- Some properties cause the Move Prover exhaust all memory and gets killed. See the "OOM"s in spec for details.

AccountScripts

This module controls whether a account can automatically accept token or not.

Formally Verified Properties

- Verified that both function never aborts and are always effective.

Authenticator

General Descriptions

Move representation of the authenticator types.

Formally Verified Properties

- In `create_multi_ed25519()`, the function aborts if threshold is zero or greater than the length of `public_keys`, and aborts if the length of `public_keys` is greater than 32.
- Converting an `authentication key` to an address assures that the resulting authentication key length is not equal to 32.
- Used `aborts_if false` to prove that function execution never aborts.
- Properties of functions that has complex implementation that is hard to formalize are given using `opaque` pragma, so that their abstract semantics can be used at callsite rather than the actual implementation.

Bitwise

This module provide bitwise operations. However, Move Prover cannot handle most of the operations.

Block

General Descriptions

Block module provide metadata for generated blocks and provide `get_current_block_number()` `get_parent_hash()` and `get_current_author()` functions to get Block metadata information.

Formally Verified Properties

- The block must exist when getting `BlockMetadata` information.
- Only genesis account can call the `process_block_metadata` function.

BlockReward

General Descriptions

The module provide block rewarding calculation logic.Provide process_block_reward to process a given block reward.

Formally Verified Properties

- Only genesis account can call `process_block_reward()` and `initialize()` .
- Checked all `aborts_if` and ensures conditions of `initialize()` function.
- Checked all `aborts_if` conditions of `process_block_reward()` .

Desired Property can't expressed with MSL, needs manual audit

- Ensures conditions of `process_block_reward()`

ChainId

General Descriptions

The module provides chain id information.

Formally Verified Properties

- Must be the genesis address to initialize ChainId. Check if ChainId exists before adding and ensure ChainId exists after adding.
- `ChainId` must exists before trying to get its details.

Collection

Verified. This deprecated module is similar to `collection2` .

Collection2

General Descriptions

This module provides a account based vector for save resource item. The resource in `CollectionStore` can be borrowed by anyone. Anyone can get immutable reference of item. Collection2 has three flags: `can_put` , `can_mut` and `can_take` , actions can only be performed if it is owner or corresponding flag is true.

Formally Verified Properties

- Verified the execution result of `push_back()` , `borrow_mut()` , `pop_back()` , `remove()` , `append()` and `append_all()` functions when `can_put` , `can_mut` or `can_take` is false.
- Verified that `CollectionStore<T>` must exist when calling `take()` , `borrow_collection()` , `destroy_empty()` or `destroy_collection()` functions.
- Verified that items field of `CollectionStore<T>` is empty when calling `destroy_empty()` or `destroy_collection()` functions.

Compare

General Descriptions

A base module that provides functions including uint and bytes comparisons.
Compare the size of the uint type value with the length of the bytes type.

Formally Verified Properties

- Verified that the correct constant is returned based on the condition in a uint size comparison.
- In the bytes comparison, due to the loop, we manually added a loop invariant at the beginning of the while block, so that Move Prover can reason about the indinces' value and proves the property that it never aborts.

Issues

- In both `cmp_bytes()` and `cmp_bcs_bytes()` , only one index is needed as indices `i1` and `i2` are always identical, and this duplication results in uneccessary gas fees.

Config

Descriptions

The `Config` is a wrapper type that holds a payload of generic type `T` , which is defined by users and can provide configuration information.

Formally Verified Properties

- Verified that addr must have the resource `Config<ConfigValue>` in `get_by_address()` , and checked the correctness of the returned value.
- When setting a configuration item to a new value, check if the signer owns the resource `ModifyConfigCapabilityHolder<ConfigValue>` , and if `ModifyConfigCapability<ConfigValue>`

is non-empty.

- Checked the existence of `Config<ConfigValue>` and `ModifyConfigCapabilityHolder<ConfigValue>` when publishing a new configuration item.
- Checked the correctness of the postcondition of updating `ModifyConfigCapabilityHolder<ConfigValue>` .

ConsensusConfig

General Descriptions

The module provide configuration of `consensus` parameters.
It mainly provides the module initialization function `initialize()` , and the function to obtain part of the configuration, `new_consensus_config()` is to create a new consensus configuration, mainly used for `DAO` .

Formally Verified Properties

- Checked the aborts conditions of all functions.
- Verified that `genesis_address` has the necessary resourcess to proceed.
- In `initialize()` , it is verfied that the acoount has to be genesis account.

ConsensusStrategy

General Descriptions

The module provides the information of current consensus strategy.

Formally Verified Properties

- Checked in `initialize()` that the blockchain must be in the genesis state and the signer must be the genesis address.
- The genesis address must have `Config<ConsensusStrategy>` and `ModifyConfigCapabilityHolder<ConsensusStrategy>` after initialization.
- Verified that `get()` returns only when `Config<ConsensusStrategy>` exists.

DAO

General Descriptions

Dao mainly includes the following parts:DaoGlobalInfo,DaoConfig,Proposal and Vote.

Formally Verified Properties

- Verified the parameter aborts of `plugin()` . Verified `DaoGlobalInfo` , `Config<DaoConfig<TokenT>>` , `ModifyConfigCapabilityHolder<DaoConfig<TokenT>>` does not exist under `sender` when `plugin()` is called.
- Verified the parameter aborts of `propose()` .Verified the impact of the existence of resources under the sender on the execution of the `propose()` .Integer overflow check for

`propose()` ignored.

- Integer overflow check for `cast_vote()` are ignored.

Math

General Descriptions

The math provide some improved math calculations

Formally Verified Properties

- Add invariant condition to while loop in sum.
- The length of the parameter nums is not 0 when calculating the mean. Integer overflow checks are ignored when summing.
- Verified that integer overflow when `mul_div()` is summed and dividend by `0`.

MerkleNFT

General Descriptions

MerkleNFT is an application designed based on MerkleTree and standard NFT protocol.

Formally Verified Properties

- Add invariant condition to while loop in `Verified()` and `register()`.
- Verified the case of `register()` integer overflow.
- Other parts verified.similar to NFT.

MintDaoProposal

General Descriptions

MintDaoProposal is a dao proposal for mint extra tokens

Formally Verified Properties

- Verified when `plugin()` adds WrappedMintCapability to address, WrappedMintCapability does not exist under address
- Verified that the `exec_delay` of `propose_mint_to()` is less than the `min_action_delay` of Dao.
- Other parts verified.similar to Dao.

ModifyDaoConfigProposal

General Descriptions

A proposal module which is used to modify Token's DAO configuration.

Formally Verified Properties

- Verified that the parameter `voting_delay` of `propose()` is greater than 100.

- Verified that the `exec_delay` of `propose()` is less than the `min_action_delay` of Dao.
- Verified that `spec_cap` is not None, the account address of spec_cap is not the same as the `sender`, and there is no `DaoConfigModifyCapability` under the sender.

NFT

General Descriptions

Non-fungible token standard and implements.Inside `NFT`, there're three optional capabilities: `MintCapability`, `BurnCapability` and `UpdateCapability`.Some functions need to verify that address has these capabilities.

Formally Verified Properties

- Verified that when calling the `register_v2()` function, `MintCapability<NFTMeta>` `BurnCapability<NFTMeta>` and `UpdateCapability<NFTMeta>` does not exist under the sender,and `NFTTypeInfoV2<NFTMeta>`, `NFTTypeInfoV2<NFTMeta>` does not exist under GENESIS_ADDRESS.
- Verified that `add_mint_capability()`, `remove_mint_capability()`, `add_burn_capability()`, `remove_burn_capability()`, `add_update_capability()` and `remove_update_capability()` parameter `sender` must have `UpdateCapability<NFTMeta>`, `MintCapability<NFTMeta>` or `BurnCapability<NFTMeta>`.
- Verified that when calling `burn()`, `mint_v2()`, `burn()` and `update_meta()`,the `sender` must have UpdateCapability
- Integer overflow when validating NFTTypeInfoV2's counter increment
- Verified that the parameter address of `get_nft_infos()`, `deposit_to()`, `count_of()`, `do_withdraw` must have `NFTGallery<NFTMeta, NFTBody>`.

OnChainConfigDao

General Descriptions

OnChainConfigDao is a DAO proposal for modify onchain configuration

Formally Verified Properties

- Verified that when `plugin ()` WrappedConfigModifyCapability is added to address, WrappedConfigModifyCapability does not exist under address
- For the execution of the `proposal()`, the proposer address has WrappedConfigModifyCapability
- Other parts verified.similar to dao.

Oracle

General Descriptions

Oracle is a bridge for smart contracts to obtain external data

Formally Verified Properties

- Verified that the `extract_signer_cap()` function must have the GenesisSignerCapability and the account address is GENESIS_ADDRESS.
- Verified that the `register_oracle()` function must have the GenesisSignerCapability::GenesisSignerCapability. `OracleInfo<OracleT, Info>` does not exist under GENESIS_ADDRESS
- Verified that `<OracleInfo<OracleT, Info>` must exist when calling the related functions `get_oracle_counter()` , `get_oracle_info()` to obtain oracle machine information, verified ensures conditions.
- Verified that `OracleFeed<OracleT, ValueT>` `DataSource<OracleT, ValueT>` and `UpdateCapability<OracleT>` do not exist when initializing the data source, `OracleInfo<OracleT, Info>` has been registered under the genesis address, and there will be no integer overflow when modifying the data source counter.
- Verified that `UpdateCapability<OracleT>` must exist under the account when calling the `update()` and `remove_update_capability()` function.
- When calling add_mint_capability remove_mint_capability add_burn_capability remove_burn_capability add_update_capability remove_update_capability, the account must have UpdateCapability, MintCapability or BurnCapability

STC

General Descriptions

STC is the standard token of Starcoin blockchain, using the `Token` infrastructure.

Formally Verified Properties

- Verified that `SharedBurnCapability` must exist when burning STC, and the value to be burned STC should be less than the total value of all the STC.
- Verified that the precision when initializing stc is less than or equal to 38.

TransactionFee

General Descriptions

TransactionFee collect gas fees used by transactions in blocks temporarily.Then they are distributed in TransactionManager.

Formally Verified Properties

- Verified that `TransactionFee<TokenType>` must exist under `GENESIS_ADDRESS` when the token is deposited into the transaction fee bucket, and there will be no integer overflow.
- Verified that `account` must be the `GENESIS_ADDRESS` during initialization, and there is no `TransactionFee<STC>` under the genesis address.

TransactionManager

General Descriptions

TransactionManager manages. prologue and epilogue of transactions. prologue of blocks.

Formally Verified Properties

- The account's auth key matches the transaction's public key.
- That the account has enough balance to pay for all of the gas.
- That the sequence number matches the transaction's sequence key.

TransactionPublishOption

General Descriptions

TransactionPublishOption provide an option to limit:whether user can use script or publish custom modules on chain

Formally Verified Properties

Verified. This deprecated module is similar to config.

TransactionTimeout

General Descriptions

TransactionTimeoutConfig whether user can use script or publish custom modules on chain.

Formally Verified Properties

- Verified the case of getting timeout timestamp integer overflow.
- Other parts verified.similar to Block.

TransactionTimeoutConfig

General Descriptions

Onchain configuration for timeout setting of transaction.

Formally Verified Properties

- Verified that the account must be `GENESIS_ADDRESS` during initialization and the blockchain is not in genesis.
- Other parts verified.similar to Block.

CoreAddresses

General Descriptions

The module provide addresses used in stdlib.

Formally Verified Properties

- Check if the signer is the genesis address in `assert_genesis_address()`.

GenesisNFT

General Descriptions

Genesis configures some NFT information and gives Genesis address related resource capabilities.

Formally Verified Properties

- Verified that the signer must be the `genesis address` to proceed with subsequent operations.
- Before adding resources to the account, you must ensure that the account does not have current resources, and after the function is executed, the account has current resources, such as `GenesisNFTMintCapability`.
- Before obtaining the resources of the genesis address through borrow_global, you must ensure that the account has current resources, such as GenesisNFTInfo.

Offer

Formally Verified Properties

- In `create()`, Verified whether the genesis address has the resource AbortsIfTimestampNotExists, and Verified whether the current timestamp plus lock_period exceeds MAX_U64.
- Verified that the signer has a resource of type <Offer> via aborts_if in create(), and ensure that it has it after the function is executed.
- Validation removes `Offer` from offer_address before having it.
- Add aborts_if false for some functions that are sure not to abort;
- Obtain the value of the resource owned by the account address through borrow_global, and add the aborts_if abort condition to ensure that the account address owns the current resource.
- In `take_offer()`, it is determined that the current time cannot be less than the time_lock of type <Offer> of offer_address.

Option

General Descriptions

This module defines the `Option` type and its methods to represent and handle an optional value.

Formally Verified Properties

- Check if Vector satisfies `is_some` and `is_none` by aborts_if.
- Ensures that the Vector has the correct value after performing operations such as deletion or addition.

- Ensure the correctness of the return value of the function through ensures.
- Use aborts_if false to express that the function will never abort.

RewardConfig

General Descriptions

The module provide configuration for block reward.

Formally Verified Properties

- Verifed the blockchain in `initialize()` must handle the creation state, the signer must be the creation address and there is no RewardConfig, and there is no Config and ModifyConfigCapabilityHolder.
- After the verification function `initialize()` is executed, it has Config and ModifyConfigCapabilityHolder.
- Verified that Config exists in the genesis address in `get_reward_config()` and `reward_delay()`, and return the correct corresponding Config.

SignedInteger64

General Descriptions

Implementation of `i64`.

Formally Verified Properties

- In `multiply_u64()`, `add_u64()` and `sub_u64()` it is guaranteed that the calculation result will not exceed MAX_U64.
- In `divide_u64()` it is guaranteed that the divisor is not 0.
- Verified the correctness of the returned value under various conditions through ensures in `sub_u64()` and `add_u64()`.

Timestamp

General Descriptions

The module implements onchain timestamp oracle. It is updated on each block and it is monotonically increasing.

Formally Verified Properties

- Verified that `initialize()`, `update_global_time()`, `set_time_has_started()` can only be called by Genesis address.
- Verified that the signer cannot have the resource `CurrentTimeMilliseconds` before `initialize()`, and there is `CurrentTimeMilliseconds` after the function is executed.
- In `update_global_time()`, verified that the genesis address must have `CurrentTimeMillis`

`econd` , and the updated timestamp must be greater than the timestamp owned by the current genesis address, which ensures the correctness of the updated post-timestamp.

- Verified that the signer must have `CurrentTimeMilliseconds` and not have `TimeHasStarted` in `set_time_has_started()` , ensure that the signer has `TimeHasStarted` after the function is executed.

Token

General Descriptions

Token implementation of Starcoin. Including registering Token, mint, burning Token, withdrawing and recharging.

Formally Verified Properties

- Verified that the precision cannot be greater than `MAX_PRECISION` in `register_token()` ,The signer must be `token_address` , and the three resources `MintCapability` , `BurnCapability` , and `TokenInfo` do not exist.
- When the registration function is executed, the signer will have three resources: `MintCapability` , `BurnCapability` , and `TokenInfo` .
- Mint capability is removed from the signer, it must have `MintCapability` and the function will no longer have `MintCapability` after execution.
- The premise of adding mint capability to signer is that there is no `MintCapability` and the function will have `MintCapability` after execution.
- The prerequisite for Verifieding that the Token destruction function is removed from the signer must have `BurnCapability` and the function will no longer have the Token destruction ability (`BurnCapability`) after the function is executed.
- Verified that the Token destruction function is added to the signer, the premise is that there is no `BurnCapability` and the function will have the Token destruction function (`BurnCapability`) after the function is executed.
- In the minting program, Verified whether the `token address` has TokenInfo and whether the total amount after minting exceeds `MAX_U128` , and the signer has `MintCapability` .
- Verified through `ensures` (`total amount = previous total + minting amount`).
- Verified that the signer burns some Tokens, the premise is that the signer has `BurnCapability` , and the burned amount cannot exceed the number of Tokens.
- Verified by `ensures` (`total amount = previous total - burn amount`).
- In the withdrawal operation, verified that the withdrawal amount cannot exceed the number of tokens you have, and that it passes the ensures verification (`total amount = previous amount - withdraw amount`).
- In the recharge operation, verified that the recharged quantity cannot exceed `MAX_U128` , and must pass the ensures verification (`total amount = previous amount + deposit amount`).

- For some functions that directly return a value, use `aborts_if false` to express that the function will never abort, and ensure the correctness of the return value through `ensures`.

Treasury

General Descriptions

The module for the Treasury of DAO, which can hold the token of DAO.

Formally Verified Properties

- Treasury can only be initialized by token_issuer, and the resource Treasury can be added to `token_issuer`.
- Get the balance to check if the `token_issuer` owns the resource `Treasury`, otherwise return 0.
- Recharge to check if the balance will exceed `MAX_U128`.
- Withdrawal check whether the withdrawal amount is 0, and the withdrawal amount cannot exceed the balance, and ensure the correctness of the balance after withdrawal.
- In `issue_linear_withdraw_capability()`, check whether period and amount are 0, and whether the genesis address has the resource CurrentTimeMilliseconds.
- In `withdraw_amount_of_linear_cap()`, the modified withdrawal is within the period and if the balance is sufficient.
- Checked whether the signer has the corresponding resource in `remove_withdraw_capability()` and `remove_linear_withdraw_capability()`.

UpgradeModuleDaoProposal

General Descriptions

A proposal module for upgrading the contract code under the token.

Formally Verified Properties

- The signer identity is the token issuer to take the next step.
- The signer does not own the `UpgradeModuleCapability` resource, and will have the `UpgradeModuleCapability` after the function is executed.

VMConfig

General Descriptions

`VMConfig` keeps track of VM related configuration, like gas schedule.

Formally Verified Properties

- `ChainId` resource must exist in the genesis address to call `gas_constants()`.
- `new_gas_cost()` never aborts, and always return a `GasCost` instance.

- To call `initialize()`, the `signer` must be the the genesis account and neither resource of type `Config` nor `ModifyConfigCapabilityHolder` exists in the genesis account.
- After `initiailize()`'s execution, both `Config` and `ModifyConfigCapabilityHolder` exist in the genesis account.

YieldFramingV2

General Description

This module defines structs that represents farming assets and user assets, as well as methods that creates yield farming pools and asset pools, stake, unstake, and related calcucation methods.

Formally Verified Properties

- Abort conditions and functional correctness of arithmetic functions
- Checked the effectiveness of operation function

Issues

It is unnecessary to check if a or b is not 0 in `div_u128()` and `mul_u128()`, which does not affect its calculation result. The extraneous `if` results in more gas consumption.

Appendix 1 - Files in Scope

This audit covered the following files:

Files	SHA-1 Hash
sources/Account.move	aae0d1359dd3b8064098b771b0850f7bcc896adb
sources/AccountScripts.move	61ef08faaa40b0a0e4e6aacf695db00c98258b25
sources/Authenticator.move	725391fc38daa73b871df563f07ca7d97cfb1ee5
sources/BCS.move	e6fa5c62aab3708b2fbc77f4085175412330dd7e
sources/Bitwise.move	ade7a9b32047c2133ca932de4161cd8542c9ca33
sources/Block.move	cab52ca92d1e8ffa9ae5d43e9123c2e691ac8a5f
sources/BlockReward.move	5f60c3fee10850ae6cbec3281da557cf3589f5e3
sources/ChainId.move	210c63f0f837509feb04bbf3bf83a69004a77373
sources/Collection.move	2e49b2730ddceb025327df9848c0940fe061184e
sources/Collection2.move	a18bb0823f0af0a0464e1dc7805e262be6648012
sources/Compare.move	6b5d618c582d980d5e27669442d7486900e4794d
sources/Config.move	ad0108f803635c58b0a93e7e950450c01dcd6783

sources/ConsensusConfig.move	5adbe8752d299a90ffee76696dcd20b88e5eab44
sources/ConsensusStrategy.move	a96ed80411d66e82164f2c557ae6aae9d1d43eee
sources/CoreAddresses.move	d014b78fe1a321bf94b51090dc456b2ca5f671ae
sources/Dao.move	237d0db3ea26a03ab8cbb95517138446859ab8d4
sources/DaoVoteScripts.move	8f7c1dd90c67fc6b96e8408ca8b722ab8bfa9293
sources/Debug.move	3372a6acf742e27ee797b269df9b2f8fe1f68b78
sources/DummyToken.move	f6da2d2ca4ccdbc7ec9baf612238ba550f9857db
sources/EmptyScripts.move	c8985db59bfafefffebed01db4fa0707a080a6ca
sources/Epoch.move	8cfcc3ae2f180264f637495f78ae059111db5bba
sources/Errors.move	27b13d04f555af2b9798820e397a9fb372d836e4
sources/Event.move	360a5d8a3f662e9e84d1e42ec26d6d84bf6f336f
sources/FixedPoint32.move	4f7cdc21f0b59bcafa3fc277651e3df5b6b95348
sources/Genesis.move	91138c5a280a2834d079df4ab5897fa144652203
sources/GenesisNFT.move	ddf2c69ee327d53a32e30ac78b9ea502da07cbbd
sources/GenesisSignerCapability.move	36b75300e7d0b90a5ba4f616856fc061839f020c
sources/Hash.move	17a4bb742062fc9016ad03df9ba69e70efb4e758
sources/LanguageVersion.move	aa070d257dde0d977e08325aed34fd29692d80c4
sources/Math.move	57f7ab5b268521294be72894e9dda3dc65344e0b
sources/MerkleNFT.move	9f4a5379d71578348a9a1518e11e6bc14d894704
sources/MintDaoProposal.move	e53f049336bbbb5c9887f22df7b52c1a4bc6d9da
sources/MintScripts.move	ed6a7aa7675362a3ecbe86d5cb1a6b8bb3ce27c9
sources/ModifyDaoConfigProposal.move	41f53c6be50f482381936400186940dc5b725447
sources/ModuleUpgradeScripts.move	dc7bfd291de644ad8db1c903e0ef11d3730cfdab
sources/NFT.move	1b2dea22a4cf346d58ea85baeb7971f2efec8ebb
sources/Offer.move	0338f329ee38a59f64d13b77968cdf9f2ac37e7e
sources/OnChainConfigDao.move	89b11a439526bbe8a927426dec8515fc34c69271
sources/OnChainConfigScripts.move	357e8b7b8621be54b1ff2defec11ba2e7ce25b8c
sources/Option.move	e8aa93c56cdda89884a7db7243726e5a87c93526
sources/Oracle.move	507cae1b0c8dbedcd658ed6dde9ff848dc654a2c
sources/PackageTxnManager.move	b3189decaf66b7594d2575c5a0c7ffafe12b7f63

sources/RewardConfig.move	15769a2d66c016c4eddac44a8831179cd68f99c8
sources/SIPs.move	e9803f819c169e25d4bb4e966fdf2f9ab26c7ead
sources/STC.move	7f3bcc7519463444b93e2dd921613ec07c2be75a
sources/SharedEd25519PublicKey.move	5d67cc1bfef8e2e672a6142098fda080955431ea
sources/Signature.move	061b82482eeb5f830375566046f7f7adcf7b38b3
sources/SignedInteger64.move	bb50c5bbdb1ce1127a94e94628eac8e314ce34f7
sources/Signer.move	056cd8a5d30cad5779db9bb86a8740407286bd63
sources/StdlibUpgradeScripts.move	eca0e6b74281d6f55b8d401054aab21bfe0a32bf
sources/Timestamp.move	f61e3a8628c429966440beec3d349d4ab3ad1350
sources/Token.move	2d6c46cb3de46f9541806b083c1779987ccc04a6
sources/TransactionFee.move	d932b28c4b9ab2a480bd737bd63a421ba24c34d9
sources/TransactionManager.move	d9ea3f976c56827f2b785d76a59a4466588a045d
sources/TransactionPublishOption.move	18f8abac5a1d425200ea5f931a19c2cab97c7daf
sources/TransactionTimeout.move	b15f7a19ef339f5c7d8d4873fcb80309d7365e76
sources/TransactionTimeoutConfig.move	dfcd7e836e92b9daa2fbc0f9b45018f40f3c2739
sources/TransferScripts.move	16ad98221915a2fe11136c709bcedb80853ce9c6
sources/Treasury.move	03b135bc339c23c09cf8956960f6522095815c7f
sources/TreasuryScripts.move	d68e0e5622fceaf735282f928b965acb1a1f22ad
sources/TreasuryWithdrawDaoProposal.move	ac668aee48ac1f46cd74d38064c27e0eac964b2b
sources/U256.move	455b949fd7393527e6095dad29a82a15fdc07b5f
sources/UpgradeModuleDaoProposal.move	cf1912ee8bd0191e457d3b0856ddacccff3e9d78
sources/VMConfig.move	d41213d9bf97b422f1203772bd47846fd60f295a
sources/Vector.move	7d48b553687f07e5201f9d9e0ed8dd39a1dab4ed
sources/Version.move	ba1f2d5c21ba48002fef1b3a0106e1451b255b14
sources/YieldFarming.move	2125b5797317d0a947c2f33e3e5b92e779594275
sources/YieldFarmingV2.move	70388bc4ca302b930d449faa09132a3793782799

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