MOVEBIT
Securing the Move Ecosystem

Mini Miners Contract
Audit Report

https://twitter.com/movebit_

contact@movebit.xyz

Mini Miners Contract Audit

MOVEBIT

1 Executive Summary

1.1 Project Information

Type	Game
Auditors	MoveBit
Timeline	2023–04–03 to 2023–04–07
Languages	Move
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	Repository: git@github.com:blocklords/sui–miner–smartcontracts.git Received Commit: b62428b74abf6b7afafde76fc137a330f466b41e Last Reviewed Commit: 9f2fcc61bfdfad82141d831174af6c0e166e5712

1.2 Issue Statistic

Item	Count	Fixed	Pending	Confirmed
Total	6	5	1	
Minor	1	1		
Medium	4	3	1	
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.
- **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

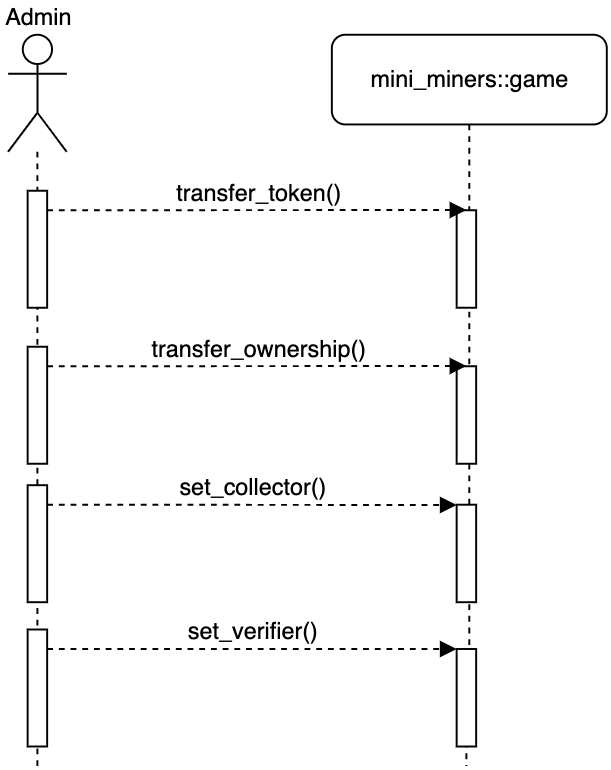
Mini Miners is a free-to-play web 3 game available on Sui BlockChain. In this fictional world, users are in charge of a mining enterprise. Users give their miners the order to travel to one of the three mines in order to extract the valuable materials required for the upkeep and expansion of user's enterprise. 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 6 issues. The audit and project teams discussed these issues together, and the project solved and confirmed all the issues.

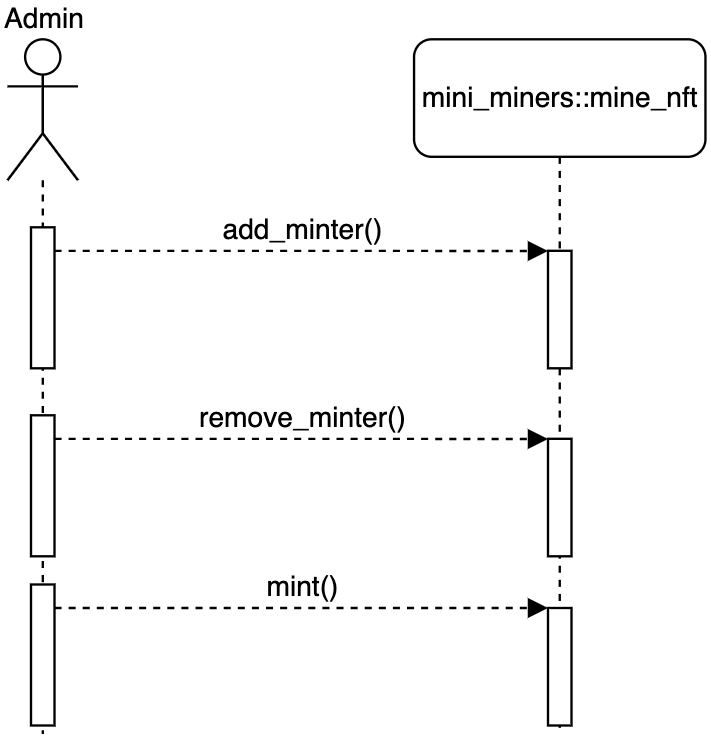
3 Participant Process

Here are the relevant actors with their respective abilities within the `Mini Miners` Smart Contract：

Admin

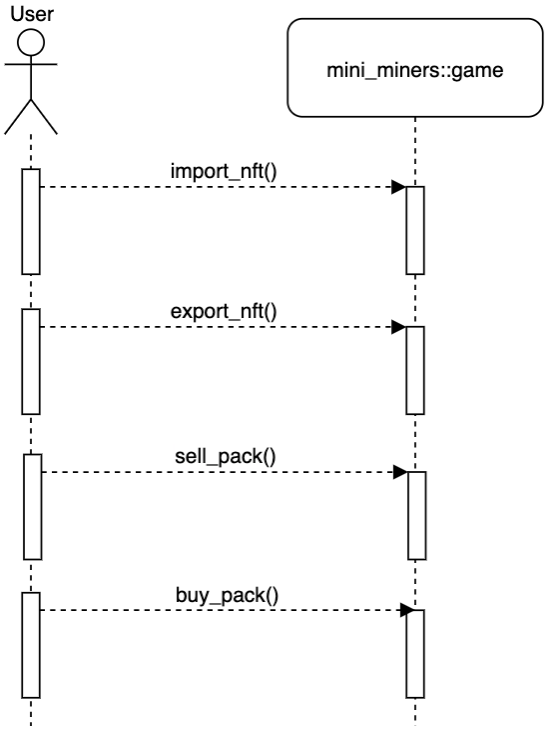
- Admin can `transfer_token`.
- Admin can `transfer_ownership`.
- Admin can `set_collector`.
- Admin can `set_verifier`.
- Admin can `add_minter`.
- Admin can `remove_minter`.
- Admin can `mint`.





User

- User can `import_nft`.
- User can `export_nft`.
- User can `sell_pack`.
- User can `buy_pack`.



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", 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

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 Compile Failed

Severity: **Medium**

Status: **Fixed**

Descriptions: The source code cannot be built. The sui-cli upgrade to 0.29 needs to update the `Move.toml` dependency, and the path of Sui Framework has changed.

Code Location: Move.toml:8.

Move.toml

```
1  [dependencies]
2  Sui = { local = "../sui/crates/sui-framework" }
3
```

Suggestion: Upgrade sui-cli to 0.29 and modify the path, also suggest using testnet just now. When publishing to mainnet, change it to mainnet.

Move.toml

```
[dependencies]
Sui = { git = "https://github.com/MystenLabs/sui.git", subdir="crates/sui-framework/packages/sui-framework/", rev = "testnet" }
```

6.2 Unbound function

Severity: Medium

Status: Fixed

Descriptions: In the latest version 0.29, Sui is updated to 0.29 `ecdsa_k1::ecrecover function` updated to `ecdsa_k1::secp256k1_ecrecover`, need to update `ecdsa_k1::ecrecover(sig, hashed_msg, hash_function)` to `ecdsa_k1::secp256k1_ecrecover(sig, msg, hash_function)`.

Code Location: sources/scdsa.move:14–52.

scdsa.move

```
public fun ecrecover_to_eth_address(signature: vector<u8>, hashed_msg: vector<u8>): address {
    // Normalize the last byte of the signature to be 0 or 1.
    let v = vector::borrow_mut(&mut signature, 64);
    if (*v == 27) {
        *v = 0;
    } else if (*v == 28) {
        *v = 1;
    } else if (*v > 35) {
        *v = (*v - 1) % 2;
    };

    let pubkey = ecdsa_k1::ecrecover(&signature, &hashed_msg);
    let uncompressed = ecdsa_k1::decompress_pubkey(&pubkey);

    // Take the last 64 bytes of the uncompressed pubkey.
    let uncompressed_64 = vector::empty<u8>();
    let i = 1;
    while (i < 65) {
        let value = vector::borrow(&uncompressed, i);
        vector::push_back(&mut uncompressed_64, *value);
        i = i + 1;
    };

    // Take the last 20 bytes of the hash of the 64-bytes uncompressed pubkey.
    let hashed = hash::keccak256(&uncompressed_64);
    let addr = vector::empty<u8>();
    let i = 12;
    while (i < 32) {
        let value = vector::borrow(&hashed, i);
        vector::push_back(&mut addr, *value);
        i = i + 1;
    };

    address::from_bytes(addr)
}
```

Suggestion: `ecrecover_to_eth_address` function needs refactoring, replace `ecdsa_k1::ecrecover with ecdsa_k1::secp256k1_ecrecover`.

6.3 Owner's address is not updated

Severity: Medium

Status: Fixed

Descriptions: `info` is a shared object, so ownership cannot be transferred through transfer, and after the change the ownership, the owner address in `info` is not updated, and the next assert will panic.

Code Location: sources/nft.move:176.

nft.move

```
public entry fun transfer_ownership(info: Info, recipient: address, ctx: &mut TxContext) {
    let sender = tx_context::sender(ctx);
    assert!(sender == info.owner, ENotOwner);

    transfer::transfer(info, recipient);

    event::emit(TransferOwnership {owner: recipient});
}
```

Suggestion: Delete the transfer and update the value of `info.owner`.

6.4 Lack check the existence of resources

Severity: Minor

Status: Fixed

Descriptions: Did not judge whether `item_id` exists before deleting.

Code Location: sources/game.move:148.

game.move

```
public fun export_nft<T: key + store>(
        game: &mut Game,
        item_id: ID,
        ctx: &mut TxContext
): T {
  let PlayerParams {
      id,
      owner,
      stake_time: _,
  } = dynamic_object_field::remove(&mut game.id, item_id);

  assert!(tx_context::sender(ctx) == owner, ENotOwner);

  let item = dynamic_object_field::remove(&mut id, true);
  object::delete(id);
  item
}
```

Suggestion: Add code: `assert!(dynamic_object_field::exists_with_type<ID, PlayerParams>(&game.id, item_id), 0);` .

6.5 PackMessage is not bound to token type

Severity: **Major**

Status: **Fixed**

Descriptions: When calculating the `message_hash` of a pack, there is only the corresponding number in the `PackMessage` structure, so when selling a pack, you can exchange for different `COIN` types, provided that the `game.id` has a corresponding Coin type.

Code Location: sources/game.move:206.

game.move

```
public entry fun sell_pack<COIN>(game: &mut Game, token_amount: u64, pack_id: u8, timestamp: u64, signature: vector<u8>, ctx: &mut TxContext) {
    let player = tx_context::sender(ctx);
    let collector = game.collector;
    ...
    let sell_gold_message = PackMessage {
          prefix: SELL_PACK_PREFIX,
          token_amount: token_amount,
          pack_id: pack_id,
          game: object::id_address(game),
          owner: player,
          timestamp: timestamp,
      };
    let message_bytes = bcs::to_bytes(&sell_gold_message);
    let message_hash = hash::keccak256(&message_bytes);
    let recovered_address = verifier::ecrecover_to_eth_address(signature, message_hash);
    assert!(game.verifier == recovered_address, ESigFail);
    assert!(dynamic_object_field::exists_<address>(&game.id, collector), ENotEnoughFunds);
    transfer::transfer(borrowed_coin, player);
    ...
}
```

Suggestion: Add the accepted token type to the `PackMessage` structure, and then calculate the hash.

game.move

```
use std::type_name::{TypeName};

struct PackMessage has drop {
    prefix: vector<u8>,
    coin_type: TypeName,
    token_amount: u64,
    pack_id: u8,
    game: address,
    owner: address,
    timestamp: u64,
}
```

6.6 Unit test can't pass

Severity: Medium

Status: Pending

Descriptions: `test_import_nft()` unit test fails.

Code Location: sources/game.move:364

game.move

```
qpb@909c4aba71a5 sui-miner-smartcontracts % sui move test
UPDATING GIT DEPENDENCY https://github.com/MystenLabs/sui.git
INCLUDING DEPENDENCY Sui
INCLUDING DEPENDENCY MoveStdlib
BUILDING MiniMiners
error[E03002]: unbound module
    ┌─ ./sources/game.move:364:35
    │
364 │             let import_nft_hash = hash::keccak256(&import_nft_byte
    s);
    │                                   ^^^^ Unbound module alias 'hash'

Failed to build Move modules: Compilation error.
```

Suggestion: import `std::hash` module.

Appendix 1 – Files in Scope

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

Files	SHA-1 Hash
./sources/nft.move	8a2b2d91a05b4b0e8a31d96763cebeff549fdf9c
./sources/ecdsa.move	bc5b506560b7a984cef4a158ac40855707598744
./sources/game.move	fcf8566662f3c525ec861540b86c924d85fec273
./Move.toml	7718f0b8cb768685cd55250cb654440471db80a8

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