Mugen Games
Smart Contract
Audit Report

MOVEBIT

contact@movebit.xyz

https://twitter.com/movebit_

06/02/2023

Mugen Games Smart Contract Audit Report

1 Executive Summary

1.1 Project Information

Description	A standard coin on Sui
Type	Token
Auditors	MoveBit
Timeline	May 22, 2023 – June 2, 2023
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://gitlab.com/Keccak256-evg/nftman/token-contracts-sui/ https://github.com/Everest-Ventures-Group/Multisig
Commits	ccc9b43cb0ecfed5f8c8361fb864f7dcf1794406 219f251ba617f7f3b6b5f0187a66c925be9eb79f 33ffdd61a86f61d049553d7085c49292d91be011 494815a1108726f262fd6b95f10e6cd21ec5951f 04d12e3486efc2982a9fb12804aa0064874120c7 b104f7380d8295b57a7cd308d8e3bacb797576e7

1.2 Files in Scope

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

ID	Files	SHA-1 Hash
MUE	Multisig/sources/multisig_example.move	4022a90e245ae50a04271f2b43126377b23cf090
MUL	Multisig/sources/multisig.move	c098a4b0b7a8014183677dc64e9af04c29824f97
ARC	token_contracts/arca/sources/arca.move	ed88d5548611917e1a94f39fd15cd3b406b8d8b6

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	15	12	3
Informational	1	1	
Minor	6	6	
Medium	3	1	2
Major	3	2	1
Critical	2	2	

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 **Mugen** to identify any potential issues and vulnerabilities in the source code of the **Mugen Games** smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified **15** issues of varying severity, listed below.

ID	Title	Severity	Status
MUL-01	`threshold` Can't Be Modified	Major	Fixed
MUL-02	Proposals Can Be Closed Before Being Executed	Major	Acknowledged
MUL-03	Useless `user` Parameter	Minor	Fixed
MUL-04	`borrow_proposal_request` Missing Permission Control	Minor	Fixed
MUL-05	`proposal_request` Can Be Extracted at Any Time	Critical	Fixed
MUL-06	Gas Optimization	Minor	Fixed
MUL-07	Assets May Be Locked In Proposals	Medium	Acknowledged
ARC-08	`burn` Function Design Flaw	Critical	Fixed
ARC-09	`Option` Params In CLI	Medium	Fixed
MUE-10	Redundant Test Code	Minor	Fixed
MUL-11	Unused Constant	Minor	Fixed
MUL-12	`approved_weight` And `reject_weight` Are Not Compared	Medium	Acknowledged
MUL-13	Unused `TxContext` Params In The Function	Informational	Fixed

MUL-14	Wrong Use Of `new_participants_by_weight`	Major	Fixed
MUL-15	Suggest Throw `abort` Instead Of Returning `false`	Minor	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the **Mugen Games** Smart Contract：

User

- User can create a `Proposal` through `create_proposal()`.
- User can change the setting of `MultiSignature` through `create_multisig_setting_proposal()` and `multisig_setting_execute()`.
- User can vote on a `Proposal` through `vote()`.
- User can close a `Proposal` through `mark_proposal_complete()`.
- User can `mint` and `burn` `ARCA` through `mint_request()` and `burn_request()`.
- User can change the `CoinMetadata` of `ARCA` through `update_metadata_request()`.

4 Findings

MUL-01 `threshold` Can't Be Modified

Severity: Major

Status: Fixed

Code Location: sources/multisig.move#L74

Descriptions: The value of `threshold` is set to 1 every time it is initialized, and there is no interface to modify the value of `threshold`, resulting in one person can control the entire `MultiSignature`, and anyone has the highest authority.

Suggestion: Add a proposal to be able to modify the value of `threshold`.

Resolution: The client confirmed the issue and fixed this issue according to the suggestion.

MUL-02 Proposals Can Be Closed Before Being Executed

Severity: Major

Status: Acknowledged

Code Location: sources/multisig.move#L159

Descriptions: Approved proposals can be marked as completed without judging whether they are executed, which may result in the approved proposal being unexecuted.

Suggestion: Make sure Proposals have been successfully executed before they can be marked as completed proposals.

Resolution: The client decides not to fix it, since execution and state are under the control of the caller's contract, and the caller needs to deal with this.

MUL-03 Useless `user` Parameter

Severity: Minor

Status: Fixed

Code Location: sources/multisig.move#L183

Descriptions: The parameter user in `pending_proposals` has no effect and cannot participate in user address filtering. It is recommended to remove it.

Suggestion: Delete the `user` parameter in the function.

Resolution: The client confirmed the issue and fixed this issue according to the suggestion.

MUL-04 `borrow_proposal_request` Missing Permission Control

Severity: Minor

Status: Fixed

Code Location: sources/multisig.move#L230

Descriptions: `borrow_proposal_request` does not add any permission restrictions, but the corresponding permission checks are done in the `is_proposal_rejected` and `is_proposal_approved` functions.

Suggestion: Add permission restrictions to make sure the user is a participant in the `MultiSignature` .

Resolution: The client confirmed the issue and fixed this issue according to the suggestion.

MUL-05 `proposal_request` Can Be Extracted at Any Time

Severity: Critical

Status: Fixed

Code Location: sources/multisig.move#L236

Descriptions: `extract_proposal_request` does not determine whether the proposal is approved or rejected. Any user of `MultiSignature` can take away the request object in the proposal.

Suggestion: Make sure to extract the `request` object after the `Proposal` has been marked as complete.

Resolution: The client confirmed the issue and fixed this issue according to the suggestion.

MUL-06 Gas Optimization

Severity: Minor

Status: Fixed

Code Location: sources/multisig.move#L279

Descriptions: The `if` of L268 can be deleted, and then the `loop` can be changed to `while` , which can improve code readability and save gas.

Suggestion: Utilizing a while loop over a range smaller than the length of the `participants_remove` vector can improve readability and reduce gas.

Resolution: The client confirmed the issue and fixed this issue according to the suggestion.

MUL-07 Assets May Be Locked In Proposals

Severity: Medium

Status: Acknowledged

Code Location: sources/multisig.move

Descriptions: A proposal cannot be canceled until it reaches an approval or rejection threshold. Proposals can get stuck if `MultiSignature` participants are inactive and not voting, causing data or assets in the proposal to be locked.

Suggestion: Add a function to cancel a `Proposal` and return the `value` object.

Resolution: The client decided not to fix it.

ARC-08 `burn` Function Design Flaw

Severity: Critical

Status: Fixed

Code Location: token_contracts/arca/sources/arca.move#L182

Descriptions: The implementation of the `burn` function is incomplete. The code cannot handle the case where the `amount` is not `none`, but it will destroy all of the coins. For example, there may be coins with a value of 10 passed in, but the `amount` is set to 3, the contract will actually destroy all coins. It is recommended to modify the `burn_request` function to split the coins before creating a new proposal, and only pass in the actual number of coins.

Suggestion: Use the `coin::split` function to separate the `coins` in params according to the `amount`.

Resolution: The client confirmed the issue and fixed this issue according to the suggestion.

ARC-09 `Option` Params In CLI

Severity: Minor

Status: Fixed

Code Location: token_contracts/arca/sources/arca.move#L147

Descriptions: The entry functions such as `burn_request` and `update_metadata_request` have Option type parameters. If the Option type parameters cannot be passed in the CLI, users

cannot call functions through CLI, which will cause DOS problems on these functions.

Suggestion: Make sure you can use the CLI and SDK to call the function or encapsulate the function.

Resolution: The client confirmed the issue and fixed this issue according to the suggestion.

MUE-10 Redundant Test Code

Severity: Minor

Status: Fixed

Code Location: sources/multisig_example.move

Descriptions: `sources/multisig_example.move` is the test code, it is recommended to delete it or move it to the test directory and add the `test_only` annotation.

Suggestion: Use `test_only` to decorate the entire module or move it to test directory.

Resolution: The client confirmed the issue and fixed this issue according to the suggestion.

MUL-11 Unused Constant

Severity: Minor

Status: Fixed

Code Location: sources/multisig.move#L10, 14

Descriptions: There are unused constants in the code such as `ERegistered`, `PROPOSAL_TYPE_MULTISIG_SETTING`, it is recommended to delete them.

Suggestion: Remove unused constants.

Resolution: The client confirmed the issue and fixed this issue according to the suggestion.

MUL-12 `approved_weight` And `reject_weight` Are Not Compared

Severity: Medium

Status: Acknowledged

Code Location: sources/multisig.move#L126,192,276

Descriptions: In the multiple execute functions of the arca module, they all use `is_proposal_approved` to judge whether the proposal is passed, The judgment condition of `is_proposal_approved` is whether the `approved_weight` is greater than the `threshold`. There is a problem in which `approved_weight` and `reject_weight` are not compared. When they both exceed the `threshold` and the `reject_weight` is greater than the `approved_weight`, is the proposal still passed?

Suggestion: It is recommended to confirm if it aligns with the design.

Resolution: The client decided not to fix it because the comparison between `approved_weight` and `reject_weight` is optional, and any one of them that reached the threshold can be executed.

MUL-13 Unused `TxContext` Params In The Function

Severity: Informational

Status: Fixed

Code Location: sources/multisig.move#L183

Descriptions: The parameter of `TxContext` type is not used in the `pending_proposals` function, which can be deleted.

Suggestion: Delete unused `_tx` parameters.

Resolution: The client confirmed the issue and fixed this issue according to the suggestion.

MUL-14 Wrong Use Of `new_participants_by_weight`

Severity: Major

Status: Fixed

Code Location: sources/multisig.move#L380

Descriptions: When getting the keys of the `participants_by_weight` vec_map, the value of `new_participants_by_weight` was got by mistake, and the keys of `participants_by_weight` should be used.

Suggestion: Use the keys from `participants_by_weight`.

Resolution: The client confirmed the issue and fixed this issue according to the suggestion.

MUL-15 Suggest Throw `abort` Instead Of Returning `false`

Severity: Minor

Status: Fixed

Code Location: sources/arca.move#L126,192,276

Descriptions: Among the multiple `execute` functions of the arca module, When the `Proposal` is not marked complete, it is recommended not to return `false`, but throw an exception abort at the end of the function, and no transaction will be generated.

Suggestion: Throw an abort at the end of the function instead of returning `false`.

Resolution: The client confirmed the issue and fixed this issue according to the suggestion.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed**: The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner

confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.