Enjoyoors

Audit Report

contact@bitslab.xyz

https://twitter.com/movebit_

Fri Feb 21 2025

Enjoyoors Audit Report

1 Executive Summary

1.1 Project Information

Description	Staking / restaking protocol WithdrawalApprover: 0x90cfa37214f0764ec6b5e70f2d088927b4b59be49333ceae0e90587e544df4ff Vault: 0x4e4c16d833abe593b81ebffb232e0798876b848fb2b55afef787c94db122edd4
Type	DeFi
Auditors	MoveBit
Timeline	Sat Feb 08 2025 - Fri Feb 21 2025
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/eq-lab/enjoyoors-sui-contracts
Commits	41f851e9e49c2760655a4bdfc7f91b9d770c2fd1 cea36acfb825589ab38ff594ce9032f350f409a2

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MOV	contracts/enjoyoors-vault/Move.toml	5a062b323e1d2a7349158253d445b201ec66ab3e
EVA	contracts/enjoyoors-vault/sources/enjoyoors_vault.move	3f1f4224f7d5616c3b3763fc5e93e941b1ba2e84
CEWAMT	contracts/enjoyoors-withdrawal-approver/Move.toml	a68982f2a60be3545c49e7c1dab1f304c6f58b60
EWA	contracts/enjoyoors-withdrawal-approver/sources/enjoyoors_withdrawal_approver.move	4ee79d5c59fa55fb0a108868b78cf8c3563974d7

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	3	3	0
Informational	1	1	0
Minor	1	1	0
Medium	1	1	0
Major	0	0	0
Critical	0	0	0

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(Optional)

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 Enjoyoors to identify any potential issues and vulnerabilities in the source code of the Enjoyoors 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 3 issues of varying severity, listed below.

ID	Title	Severity	Status
EVA-1	Missing Event for Role Removal in `grant_role` Function	Minor	Fixed
EVA-2	Ensure `delta > 0` in Supply Limit Modifications	Informational	Fixed
EWA-1	Single-step Ownership Transfer Can be Dangerous	Medium	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the Enjoyoors Smart Contract :

Admin

- `grant_role` : Assigns a specific role to an address.
- `add_coin` : Adds a new coin to the vault and initializes its deposit storage.
- `change_approval_table_id` : Updates the approval table ID for withdrawals.
- `change_min_deposit` : Modifies the minimum deposit amount for a specific coin.
- `increase_supply_limit` : Increases the supply limit of a specific coin.
- `decrease_supply_limit` : Decreases the supply limit of a specific coin.
- `pause_deposit` : Pauses deposits for a specific coin.
- `pause_withdrawal` : Pauses withdrawals for a specific coin.
- `pause_claim` : Pauses claims for a specific coin.
- `resume_deposit` : Resumes deposits for a specific coin.
- `resume_withdrawal` : Resumes withdrawals for a specific coin.
- `resume_claim` : Resumes claims for a specific coin.
- `change_withdrawal_period` : Updates the withdrawal approval period.
- `transfer_admin` : Current admin designates a new pending admin address.
- `cancel_admin_transfer` : Cancels an ongoing admin rights transfer process.
- `accept_admin` : Pending admin accepts the admin role.

User

- `deposit` : Deposits coins into the vault.
- `request_withdrawal` : Requests a withdrawal of a specific amount of coins.
- `finalize_withdrawal` : Completes a withdrawal after approval.
- `claim_withdrawal` : Approves and finalizes a withdrawal request after the waiting period.

4 Findings

EVA-1 Missing Event for Role Removal in `grant_role` Function

Severity: Minor

Status: Fixed

Code Location:

contracts/enjoyoors-vault/sources/enjoyoors_vault.move#383

Descriptions:

```
    public fun grant_role(config: &mut VaultConfig, role: u8, to: address, ctx: &mut
TxContext) {
        only_role(config, ADMIN_ROLE, ctx.sender());
        assert!(role < ROLES_COUNT, EInvalidRoleId);

        let actual = config.roles.try_get(&role);

        if (actual.is_some()) {
            config.roles.remove(&role);
        };

        config.roles.insert(role, to);

        event::emit(RoleGranted {
            role: role,
            account: to,
            sender: ctx.sender(),
        });
    }
```

In the `grant_role` function, when assigning a new role, the existing role is first removed before the new assignment takes place. However, there is no event emitted to indicate that a role was removed. This lack of logging can make it difficult to track role changes.

Suggestion:

Emit a `RoleRevoked` event before removing the existing role.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

EVA-2 Ensure `delta > 0` in Supply Limit Modifications

Severity: Informational

Status: Fixed

Code Location:

contracts/enjoyoors-vault/sources/enjoyoors_vault.move#472,489

Descriptions:

```
public fun increase_supply_limit(
    config: &mut VaultConfig,
    coin_metadata_id: ID,
    delta: u64,
    ctx: &mut TxContext
) {
    only_role(config, SETUP_ROLE, ctx.sender());

    let coin_config = config.coins.borrow_mut(coin_metadata_id);
    coin_config.supply_till_limit = coin_config.supply_till_limit + delta;

    event::emit(SupplyLimitIncreased {
        coin_metadata_id: coin_metadata_id,
        delta: delta,
    })
}

public fun decrease_supply_limit(
    config: &mut VaultConfig,
    coin_metadata_id: ID,
    delta: u64,
    ctx: &mut TxContext
) {
    only_role(config, SETUP_ROLE, ctx.sender());
    let coin_config = config.coins.borrow_mut(coin_metadata_id);
    assert!(coin_config.supply_till_limit >= delta, ESupplyLimitDecreaseFailed);
    coin_config.supply_till_limit = coin_config.supply_till_limit - delta;

    event::emit(SupplyLimitDecreased {
        coin_metadata_id: coin_metadata_id,
        delta: delta,
```

```
    })
}
```

In the functions `increase_supply_limit` and `decrease_supply_limit`, the parameter delta represents the amount by which the supply limit is adjusted. However, there is no validation to ensure that delta > 0.

Suggestion:

Add an assertion to ensure that delta > 0, ensuring meaningful updates.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

EWA-1 Single-step Ownership Transfer Can be Dangerous

Severity: Medium

Status: Fixed

Code Location:

contracts/enjoyoors-withdrawal-approver/sources/enjoyoors_withdrawal_approver.move#122;
contracts/enjoyoors-vault/sources/enjoyoors_vault.move#376

Descriptions:

The `change_admin()` and `grant_role` function has a problem with single-step ownership permission transfer.

Single-step ownership transfer means that if a wrong address was passed when transferring ownership or admin rights it can mean that role is lost forever. If the admin permissions are given to the wrong address within this function, it will cause irreparable damage to the contract.

Suggestion:

It is recommended to adopt a two-step administrator transfer process:

- Nomination: The current administrator nominates the new administrator.
- Acceptance: The transfer is officially completed once the nominated address accepts the administrator role.

By requiring the new administrator to explicitly accept the role, the risk of unintended or mistaken transfers is minimized.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

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.