Supra VRF

Audit Report

MOVEBIT

contact@movebit.xyz

https://twitter.com/movebit_

Mon Nov 06 2023

Supra VRF Audit Report

1 Executive Summary

1.1 Project Information

Description	Trustless randomness, with proof you can count on
Type	Oracle
Auditors	MoveBit
Timeline	Mon Oct 16 2023 – Mon Nov 06 2023
Languages	Move
Platform	Aptos
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/Entropy-Foundation/supra-vrf
Commits	0a607a60fcfd2adc59a680604b6d11e9d96aa821 916753e4551b4cdbc9421dfd20c23169e36e2059 bedb9a4820b22670096e0ea8766a65aaccf1e727

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MOV	smart-contracts/aptos/Move.toml	e71cb058b2eb82d57c7bae77ed3ff0e35fdc28d7
SUT	smart-contracts/aptos/sources/supra_util.move	90b9fee52e4ef725b67c22835c96cbb8dad6b141
FNO	smart-contracts/aptos/sources/free_node.move	5e3123feef170311e062c66d296d1fb7d4cb4999
SVR	smart-contracts/aptos/sources/supra_vrf.move	1f95ccbfef7f7a46c8571fc42ffcb754b1c13bf8
DEP	smart-contracts/aptos/sources/deposit.move	0061dc9e28af160e72c4c7fbd7c60e6cf93b8cae

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	7	6	1
Informational	0	0	0
Minor	2	2	0
Medium	1	1	0
Major	4	3	1
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

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 Supra to identify any potential issues and vulnerabilities in the source code of the Supra VRF 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 7 issues of varying severity, listed below.

ID	Title	Severity	Status
DEP-1	Lack of Contract Status Verification	Major	Fixed
DEP-2	Removing Client Address Results In Funds Being Locked	Major	Fixed
DEP-3	Improper Resource Management	Major	Fixed
DEP-4	Zero-Balance Accounts Not Removed	Medium	Fixed
DEP-5	Unused Constant	Minor	Fixed
DEP-6	Unused Events	Minor	Fixed
SVR-1	Centralization Risk	Major	Acknowledged

3 Participant Process

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

Admin

- Admin can set the minimum balance limit for funds through the function `set_minbalance_limit_supra()`.
- Admin can set the contract status through `set_contract_disabling()`.
- Admin can claim node expenses through `claim_free_node_expenses()`.
- Admin can collect transaction fees from clients through the function `add_client_address()`.
- Admin can add clients to the whitelist through the function `add_client_address()`.
- Admin can remove the customer from the whitelist through the function `remove_client_address()`.
- Admin can remove all contracts for the customer through the function `remove_all_client_contracts`.
- Admin can update client subscription end date through the function `update_client_subscription()`.

Client

- The client can set the minimum balance limit through the function `client_setting_minimum_balance()`.
- The client can whitelist the contract address through the function `add_contract_to_whitelist()`.
- The client can remove the whitelisted contract address through the function `remove_contract_from_whitelist()`.
- The client can deposit tokens through the function `deposit_fund()`.
- The client can withdraw tokens through the function `withdraw_fund()`.

Owner

- The owner can add a free-node address to the whitelist through the function `add_whitelist()`.

- The owner can add free-node addresses in bulk into the whitelist through the function `add_whitelist_bulk()`.
- The owner can remove the free-node address from the whitelist through the function `remove_whitelist()`.
- The owner can remove free-node addresses in bulk from the whitelist through the function `remove_whitelist_bulk()`.
- The owner can update `public_key` through the function `update_public_key()`.

4 Findings

DEP–1 Lack of Contract Status Verification

Severity: Major

Status: Fixed

Code Location:

smart–contracts/aptos/sources/deposit.move#349,369

Descriptions:

If the administrator has already suspended operations on `DepositManagement`, the functions `add_contract_to_whitelist()` and `remove_contract_from_whitelist()` should verify the current contract status before allowing the addition or removal of contract addresses. Failing to do so may result in unexpected behavior and potentially serious vulnerabilities.

Suggestion:

It is recommended to add a status check.

Resolution:

This issue has been fixed. The client added a status check.

DEP-2 Removing Client Address Results In Funds Being Locked

Severity: Major

Status: Fixed

Code Location:

smart-contracts/aptos/sources/deposit.move#277-294

Descriptions:

In the `remove_client_address` function, removing a client's address is a straightforward removal from the whitelist, which can potentially result in the client's funds being permanently locked. If the client has a remaining balance, this portion of the funds becomes irretrievable. The only way to unlock this portion of funds is by re-adding the user's address to the whitelist.

```
    public entry fun remove_client_address(sender: &signer, client_address: address)
acquires DepositManagement, DepositEvents {
        ensure_supra_admin(sender);
        ensure_contract_enabled();
        ensure_whitelisted_client(client_address);

        // removing from whitelisted table
        let deposit_management = borrow_global_mut<DepositManagement>
(@supra_addr);
        table::remove(&mut deposit_management.whitelist_clients, client_address);
        table::remove(&mut deposit_management.subscription_period, client_address);

        // Emit and event that admin has removed client wallet address from whitelist
        let event_handler: &mut DepositEvents = borrow_global_mut<DepositEvents>
(@supra_addr);
        emit_event<ClientRemoveEvent>(
            &mut event_handler.client_remove_from_whitelist_events,
            ClientRemoveEvent { client_address }
        );
    }
```

The way clients can withdraw their balances is through the `withdraw_fund` function. If this record is deleted, clients will experience a failed withdrawal, and moreover, admin will also be unable to access these funds. Admin extract funds using the `claim_free_node_expenses` function. However, directly deleting the client's address without updating the `balance_management.supra_fund` value leads to a scenario where the client's funds remain permanently locked in the resource account.

```
    public entry fun claim_free_node_expenses(sender: &signer, receiver_address:
address, amount: u64) acquires BalanceManagement {
        ensure_multisig_account(sender);

        let resource_address = account::create_resource_address(&@supra_addr,
SEED_BALANCE);
        let balance_management = borrow_global_mut<BalanceManagement>
(resource_address);

        ensure_enough_balance(balance_management.supra_fund, amount);
        balance_management.supra_fund = balance_management.supra_fund - amount;

        let resource_signer =
account::create_signer_with_capability(&balance_management.signer_cap);
        coin::transfer<AptosCoin>(&resource_signer, receiver_address, amount);
    }
```

Suggestion:

It is recommended to check if the client's balance is zero when deleting their address (which is highly unlikely due to the min balance limit) or to transfer the remaining funds to the user when removing the client's address.

Resolution:

This issue has been fixed. The client will transfer funds to the user or `supra_fund` when removing the user's address.

DEP-3 Improper Resource Management

Severity: Major

Status: Fixed

Code Location:

smart-contracts/aptos/sources/deposit.move#410-426

Descriptions:

If the user's `client_balance` is already 0, the code doesn't remove it from the `balance_management.client_balance`.

```
 public entry fun withdraw_fund(sender: &signer, withdraw_amount: u64) acquires
BalanceManagement, DepositManagement {
        let client_addresss = signer::address_of(sender);
        ensure_contract_enabled();
        ensure_whitelisted_client(client_addresss);

        let resource_address = account::create_resource_address(&@supra_addr,
SEED_BALANCE);
        let balance_management = borrow_global_mut<BalanceManagement>
(resource_address);

        let resource_signer =
account::create_signer_with_capability(&balance_management.signer_cap);
        let client_balance_mut = table::borrow_mut(&mut
balance_management.client_balance, client_addresss);
        let client_balance = *client_balance_mut;

        ensure_enough_balance(client_balance, withdraw_amount);

        *client_balance_mut = client_balance - withdraw_amount;
        coin::transfer<AptosCoin>(&resource_signer, client_addresss, withdraw_amount);
    }
```

Suggestion:

It is recommend to remove the client from `balance_management.client_balance` when user's `client_balance` is 0.

Resolution:

This issue has been fixed. If the client's balance is zero after users withdraw their funds, the client has already been removed from `balance_management.client_balance`.

DEP-4 Zero-Balance Accounts Not Removed

Severity: Medium

Status: Fixed

Code Location:

smart-contracts/aptos/sources/deposit.move#205-225

Descriptions:

In the `collect_tx_fee_from_client()` function, the protocol calls the `deposit.collect_fund()` function to collect funds into `supra_fund`. As shown in the code snippet below, the protocol deducts the withdrawal amount from the client's balance and updates the balance in the table.

```
  let client_balance_mut = table::borrow_mut(&mut balance_management.client_balance,
client_address);
        let client_balance = *client_balance_mut;
        ensure_enough_balance(client_balance, withdraw_amount);

        *client_balance_mut = client_balance - withdraw_amount;
        balance_management.supra_fund = balance_management.supra_fund +
withdraw_amount;
```

In edge cases where the client's balance is reduced to zero, the protocol does not remove the client from the `balance_management.client_balance` table.

Suggestion:

It is recommended to remove the client from `balance_management.client_balance` when the user's `client_balance` is 0.

Resolution:

This issue has been fixed. If the client's balance is zero, the client has already been removed from `balance_management.client_balance`.

DEP-5 Unused Constant

Severity: Minor

Status: Fixed

Code Location:

smart-contracts/aptos/sources/deposit.move#35-38

Descriptions:

These two error value constants are not used in the contract, `EACCOUNT_NOT_FOUND` , `EACCOUNT_NOT_REGISTERED_FOR_APT` .

```
/// Account does not exist.
const EACCOUNT_NOT_FOUND: u64 = 8;
/// Account is not registered to receive APT.
const EACCOUNT_NOT_REGISTERED_FOR_APT: u64 = 9;
```

Suggestion:

It is recommended to remove the unused constant.

Resolution:

This issue has been fixed. The client has already used these constants.

DEP-6 Unused Events

Severity: Minor

Status: Fixed

Code Location:

smart-contracts/aptos/sources/deposit.move#76,103

Descriptions:

The `supra_refunded_events` event is defined but not triggered, resulting in unnecessary gas consumption. Additionally, the `SupraRefunded` structure is redundant, further contributing to the excess gas consumption.

Suggestion:

It is recommended to delete the unused event and structure.

Resolution:

This issue has been fixed. The client deleted the unused event and structure.

SVR-1 Centralization Risk

Severity: Major

Status: Acknowledged

Code Location:

smart-contracts/aptos/sources/supra_vrf.move#108-120

Descriptions:

- The Admin can set `min_balance_limit_supra` through `set_minbalance_limit_supra()`.
- The Admin can add client wallet account address in to whitelist through `set_contract_disabling()`.
- The Admin can add client wallet account address in to whitelist through `add_client_address()`.
- The Admin can remove client address through `remove_client_address()`.
- The Admin can remove all contract address from client's whitelist through `remove_all_client_contracts()`.
- The Admin can update client subscription end date through `update_client_subscription()`.
- The Admin can add free-node address into whitelist through `add_whitelist()`.
- The Admin can add free-node addresses in bulk into whitelist through `add_whitelist_bulk()`.
- The Admin can remove free-node address from whitelist through `remove_whitelist()`.
- The Admin can update public key through `update_public_key()`.

Suggestion:

It is recommended to take some measures to mitigate centralization risk.

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