KGeN Swap

Audit Report

contact@bitslab.xyz

https://twitter.com/movebit_

Tue Aug 26 2025

KGeN Swap Audit Report

1 Executive Summary

1.1 Project Information

Description	A token exchange contract in which users exchange input tokens for output tokens at a fixed ratio
Type	DeFi
Auditors	MoveBit
Timeline	Tue Aug 12 2025 - Tue Aug 26 2025
Languages	Move
Platform	Aptos
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/kgen-protocol/smartcontracts
Commits	ec2b2fe424541cd7250e02d11e60ec04d2386fd7 535bdbdb3b87f40270d45c34e21bf1fdfa13808d

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MOV11	rKGeN-Swap/Move.toml	72b7098c1509648366b7e7209e94d4f3102eb39b
RKG	rKGeN-Swap/sources/rkgen.move	537fa6b8b7c176e65a9043613eccd2e1208f00da

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	10	10	0
Informational	4	4	0
Minor	2	2	0
Medium	4	4	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 KGeN to identify any potential issues and vulnerabilities in the source code of the KGeN Swap 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 10 issues of varying severity, listed below.

ID	Title	Severity	Status
RKG-1	Single-step Ownership Transfer Can be Dangerous	Medium	Fixed
RKG-2	Incorrect Accounting of Collected Fees in `swap_sponsor()` Function	Medium	Fixed
RKG-3	`update_swap_fee_rate` Parameter Range Error	Medium	Fixed
RKG-4	`get_sponsor_swap_preview` Fee Rounded Up	Medium	Fixed
RKG-5	Potential Overflow in `get_swap_preview()` Calculation Due to Large `amount_in`	Minor	Fixed
RKG-6	Call `set_untransferable()` to Prevent the Second Store from being Transferable	Minor	Fixed
RKG-7	Missing Validation for Identical Tokens in `create_pool()` Function	Informational	Fixed

RKG-8	Incorrect Comment in `swap()` Function for Token Transfer Description	Informational	Fixed
RKG-9	`SwapPauseStatisChanges` Spelling Mistake	Informational	Fixed
RKG-10	`get_sponser_swap_preview` Spelling Mistake	Informational	Fixed

3 Participant Process

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

Admin

- `pause_swap` : Suspend or resume the swap function of the contract
- `create_pool` Create an exchange pool between input tokens and output tokens, and set the initial rate, exchange rate, and fee receiving address
- `swap_sponsor` Redemption where the administrator pays the Gas fee on behalf of the user. Input tokens are exchanged for output tokens. The input tokens are transferred to the destruction address, and the output tokens (after deducting the exchange fee and Gas fee) are transferred to the user
- `update_admin` : Update the administrator address
- `update_swap_fee_rate` : Update the exchange handling fee rate
- `update_swap_ratio` : Update exchange rate
- `update_fee_recipient` : Update the address of the recipient of the handling fee
- `deposit` : Deposit output tokens into the pool to increase liquidity
- `withdraw` : Extract output tokens from the pool

Users

- `swap` : Users exchange input tokens for output tokens. Input tokens are transferred to the destruction address, output tokens are transferred from the pool to the user, and transaction fees are transferred to the fee recipient

4 Findings

RKG-1 Single-step Ownership Transfer Can be Dangerous

Severity: Medium

Status: Fixed

Code Location:

rKGeN-Swap/sources/rkgen.move#464-476

Descriptions:

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.

```
public entry fun update_admin(admin: &signer, new_admin: address) acquires  Admin {
    assert_admin(admin);

    let admin_resource = borrow_global_mut<Admin>(@rkgen);
    let old_admin = admin_resource.admin;
    admin_resource.admin = new_admin;

    event::emit(AdminUpdated{
       old_admin,
       new_admin,
       updated_by: signer::address_of(admin),
    });
  }
```

The `update_admin()` function directly updates the admin to a new address, which introduces a security risk.

Suggestion:

It is recommended to implement a two-step admin transfer process instead of directly updating the admin.

Resolution:

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

RKG-2 Incorrect Accounting of Collected Fees in `swap_sponsor()` Function

Severity: Medium

Status: Fixed

Code Location:

rKGeN-Swap/sources/rkgen.move#451

Descriptions:

In the `swap_sponsor()` function, the protocol transfers `total_fee_amount` to the `fee_recipient`.

```
    // Transfer fee and gas_fee to fee recipient (in output token)
    if(total_fee_amount > 0){
       dispatchable_fungible_asset::transfer(
          output_token_store_signer,
          output_token_store,
          primary_fungible_store::primary_store(pool.fee_recipient,
pool.output_token_metadata),
           total_fee_amount
       );
    };
```

However, the accounting variable `total_fees_collected` only records `swap_fee_amount`, leading to inconsistent fee tracking.

```
    pool.total_input_token_swapped += amount;
    pool.total_output_token_swapped += amount_out;
    pool.total_fees_collected += swap_fee_amount;
```

Suggestion:

It is recommended to update the implementation so that `total_fees_collected` records the full `total_fee_amount` instead of only `swap_fee_amount`.

Resolution:

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

RKG-3 `update_swap_fee_rate` Parameter Range Error

Severity: Medium

Status: Fixed

Code Location:

rKGeN-Swap/sources/rkgen.move#478

Descriptions:

When setting update_swap_fee_rate, new_swap_fee_rate is allowed to be less than or equal to MAX_FEE_RATE. However, when it is set to exactly equal, In the 'get_sponser_swap_preview' function (called by 'swap_sponsor'), there exists an assertion 'assert! (total_fee_amount < output_amount, EFEES_EXCEED_AMOUNT) '. If the administrator sets' swap_fee_rate 'to its maximum allowable value of' 10,000 '(i.e., 100%), this assertion will definitely fail for any call. Because when the rate is 100%, 'swap_fee_amount' will equal 'output_amount'. Since 'total_fee_amount' also contains' swap_gas_fee_amount 'greater than zero,' total_fee_amount 'will always be greater than' output_amount ', causing the transaction to roll back. This makes the 'swap_sponsor' feature completely disabled under specific (but valid) parameter configurations.

```
    public entry fun update_swap_fee_rate(admin: &signer, new_swap_fee_rate: u64)
acquires Admin, SwapPool {
        assert_admin(admin);
        assert!(new_swap_fee_rate <= MAX_FEE_RATE, EINVALID_FEE_RATE);
        assert_pool();

        let pool = borrow_global_mut<SwapPool>(@rkgen);
        let old_swap_fee_rate = pool.swap_fee_rate;
        pool.swap_fee_rate = new_swap_fee_rate;

        event::emit(SwapFeeRateUpdated{
            old_swap_fee_rate,
            new_swap_fee_rate,
```

```
            updated_by: signer::address_of(admin),
        });
    }
```

Suggestion:

Modify to

```
assert!(new_swap_fee_rate < MAX_FEE_RATE, EINVALID_FEE_RATE);
```

Resolution:

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

RKG-4 `get_sponsor_swap_preview` Fee Rounded Up

Severity: Medium

Status: Fixed

Code Location:

rKGeN-Swap/sources/rkgen.move#251

Descriptions:

The revenue and fee calculation of the swap_fee_amount protocol should be rounded down, while the fee portion collected by the protocol should be rounded up to avoid loss of accuracy and resulting in damage to the contract's revenue

```
    public fun get_sponser_swap_preview(amount_in: u64, gas_fee_amount: u64): (u64, u64, u64) acquires  SwapPool {
        assert_amount(amount_in);
        assert!(gas_fee_amount > 0, EINVALID_GAS_FEE);
        assert_pool();
        let pool = borrow_global<SwapPool>(@rkgen);

        // Calculate output amount based on swap ratio
        let output_amount = (amount_in * pool.swap_ratio) / FEE_RATIO_PRECISION;

        // Calculate swap fee and gas fee on the output amount
        let swap_fee_amount = (output_amount * pool.swap_fee_rate) / FEE_PRECISION;
        let total_fee_amount = swap_fee_amount + gas_fee_amount;
        assert!(total_fee_amount < output_amount, EFEES_EXCEED_AMOUNT);
        let amount_out = output_amount - total_fee_amount;

        (amount_out, swap_fee_amount, total_fee_amount)
    }
```

Suggestion:

Modify it to round up

Resolution:

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

RKG-5 Potential Overflow in `get_swap_preview()` Calculation Due to Large `amount_in`

Severity: Minor

Status: Fixed

Code Location:

rKGeN-Swap/sources/rkgen.move#241

Descriptions:

The function `get_swap_preview()` is used to calculate the swap output amount. The `swap_ratio_amount` is calculated as:

```
swap_ratio_amount = (amount_in * pool.swap_ratio) / FEE_RATIO_PRECISION
```

where `0 < swap_ratio < 10000`, `amount_in` has 8 decimals, and `max(u64) ≈ 1e19 < 1e20`. Assuming `swap_ratio` has 3 decimals and `amount_in` has 8 decimals, when `amount_in` is very large, this calculation may lead to an **overflow**.

Suggestion:

It is recommended to convert the variables to `u128` before performing the multiplication to prevent overflow during calculation.

Resolution:

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

RKG-6 Call `set_untransferable()` to Prevent the Second Store from being Transferable

Severity: Minor

Status: Fixed

Code Location:

rKGeN-Swap/sources/rkgen.move#293-325

Descriptions:

In the `create_pool()` function, the protocol creates a second store for fungible assets. By default, this second store is transferable, which may lead to unintended asset transfers or security issues.

```
    // Create store for output token only
        let output_token_store_constructor= &object::create_object(@rkgen);

fungible_asset::create_store(output_token_store_constructor,output_token_metadata);
        let output_token_store_extend_ref =
object::generate_extend_ref(output_token_store_constructor);
```

Suggestion:

It is recommended to call the `set_untransferable()` function after creating the second store to prevent it from being transferable, avoiding potential problems.

Resolution:

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

RKG-7 Missing Validation for Identical Tokens in `create_pool()` Function

Severity: Informational

Status: Fixed

Code Location:

rKGeN-Swap/sources/rkgen.move#293-325

Descriptions:

The `create_pool()` function is used to create a swap pool between the input token and output token.

```
  public entry fun create_pool(admin: &signer, input_token_metadata:
Object<Metadata>, output_token_metadata: Object<Metadata>, initial_fee_rate: u64,
initial_swap_ratio: u64, fee_recipient: address) acquires  Admin {
        assert_admin(admin);
        assert!(initial_fee_rate <= MAX_FEE_RATE, EINVALID_FEE_RATE);
        assert_swap_ratio(initial_swap_ratio);
        assert!(!exists<SwapPool>(@rkgen), EPOOL_NOT_EXISTS);

        // Create store for output token only
        let output_token_store_constructor= &object::create_object(@rkgen);

fungible_asset::create_store(output_token_store_constructor,output_token_metadata);
        let output_token_store_extend_ref =
object::generate_extend_ref(output_token_store_constructor);
```

However, the protocol does not validate that `input_token != output_token`, which could allow the creation of an invalid pool.

Suggestion:

It is recommended to add a check to ensure that `input_token` and `output_token` are not the same before creating the pool.

Resolution:

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

RKG-8 Incorrect Comment in `swap()` Function for Token Transfer Description

Severity: Informational

Status: Fixed

Code Location:

rKGeN-Swap/sources/rkgen.move#355

Descriptions:

The `swap()` function is used to swap the input token for the output token at the swap ratio. However, when transferring the input token from the user to the burn vault address, the comment incorrectly states "Transfer output token from user to admin". This is misleading and does not reflect the actual logic.

```
    // Transfer output token from user to admin
     dispatchable_fungible_asset::transfer(
        user,
        primary_fungible_store::primary_store(user_addr, pool.input_token_metadata),
        primary_fungible_store::primary_store(burn_vault_address,
pool.input_token_metadata),
        amount
     );
```

Suggestion:

It is recommended to update the comment to correctly describe the action as "Transfer input token from user to burn vault address" to ensure clarity and avoid confusion for future maintenance and audits.

Resolution:

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

RKG-9 `SwapPauseStatisChanges` Spelling Mistake

Severity: Informational

Status: Fixed

Code Location:

rKGeN-Swap/sources/rkgen.move#138

Descriptions:

There is a spelling mistake in the naming of the SwapPauseStatisChanges structure. It should be SwapPauseStatusChanges

Suggestion:

Correct the incorrect naming

Resolution:

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

RKG-10 `get_sponser_swap_preview` Spelling Mistake

Severity: Informational

Status: Fixed

Code Location:

rKGeN-Swap/sources/rkgen.move#251

Descriptions:

There is a spelling mistake in the function name get_sponser_swap_preview. The correct function name should be get_sponsor_swap_preview`

Suggestion:

Correct the incorrect function name

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.