Atmos Smart Contract

Audit Report

MOVEBIT

contact@bitslab.xyz

https://twitter.com/movebit_

Fri Mar 07 2025

Atmos Smart Contract Audit Report

1 Executive Summary

1.1 Project Information

Description	The liquidity engine and native Super DEX for SUPRA_Labs.
Type	DeFi
Auditors	MoveBit
Timeline	Fri Feb 07 2025 - Fri Mar 07 2025
Languages	Move
Platform	Others
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/AtmosDex/atmos-mainnet-contracts
Commits	1be7ab9dc77416de0a752879141f38e764d3e04c 39d9e6821375e99b1999f57a98300d82f71819b7 cbdfe673485189540d9288d2754698b13f9e1419 feb8500dcd8c4a8d2f6f326cc3c34b3113effcd4 c7c0a135d0b0b3c30b20527c7e8df59b2586c62c 0c8ea641c16316a5ae115670d19988111f14584b eab5f8d5fae5a774963f8fe1f985cce0b6293eb7 b9f5e567b0c3fde5443cf8248972b80f74cd9ade

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MOV	Math/Move.toml	71ba0a7f8961af3f5f450ff08149f21e5dc4f903
SMA	Math/sources/stable_math.move	7099f2a7f4948f5b8624891c4f5b3eacdb6229c2
WMA	Math/sources/weighted_math.move	79433dd7d8938adcfca9ed2f7b37ca6b1081f844
MHE	Math/sources/math_helper.move	b29e34d75b39bcb6fc98943ca47b7b0712f50453
MOV1	Move.toml	4a40abf34395f7b8ad6829779851647d31cdbdea
ENT	sources/entry.move	70d3fbd94a4fe651d9c3f753701357de28d71264
ACL	sources/acl.move	c76a3ec032765aa9d4332b314a7da0355b15382d
EVE	sources/events.move	80f1731b8c4a133bcb84d3764c00baf537d44362
GCO	sources/global_config.move	97f9d89c3f978452c8d85911842928464f574e95
CUT	sources/coin_utils.move	97decbfd93c58a5b59b695ac17290809fd56eb35
LPO	sources/liquidity_pool.move	52edaf97a4cdd62d609f08cc37fb5ecfb1c551c8

LAU	sources/launchpad.move	f7d7914b3b421595943724fbbc060aa0134dd470
CEN	sources/coin_entry.move	9a1c1d9493bcc856b8a497eb0c78d6458d6e64fe
TRE	sources/treasury.move	d74987767ef5f29981c03a50b2685373f6f424c9
RAC	sources/resource_account.move	d47f7b7a8676b1c74d52db771bd197c848c753f7
LOR	sources/limit_orders.move	6724d73805818401e73909e3cc42b777cbd124b0
DCA	sources/dca.move	6545fdef13e09b86640041111cef25895cf2f068

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	21	19	2
Informational	3	3	0
Minor	5	5	0
Medium	9	7	2
Major	3	3	0
Critical	1	1	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 Atmos to identify any potential issues and vulnerabilities in the source code of the Atmos 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 21 issues of varying severity, listed below.

ID	Title	Severity	Status
CUT-1	Incorrect Check in `deposit_by_version()`	Major	Fixed
DCA-1	Bounds Check Error	Medium	Fixed
DCA-2	Single Failure Will Block Subsequent Operations	Medium	Acknowledged
DCA-3	Stored Fungible Assets are Withdrawn by Users	Medium	Acknowledged
ENT-1	`pool_types` Value Error	Medium	Fixed
GCO-1	Code Logic Flaws	Minor	Fixed
GCO-2	Code Readability Issues	Informational	Fixed
GCO-3	Unused Constant	Informational	Fixed
LAU-1	Faulty Sell Function Logic Enables Free SUPRA Acquisition	Critical	Fixed
LAU-2	The Input Parameters are Incorrect	Major	Fixed
LAU-3	No Refund Logic	Medium	Fixed

LAU-4	Duplicate Platform Fee Charge in the `buy` Function	Medium	Fixed
LAU-5	Accuracy Issues	Minor	Fixed
LOR-1	Multiple Execution Issues	Medium	Fixed
LPO-1	`fee_bps` Validation Error	Medium	Fixed
LPO-2	Loss of Precision	Minor	Fixed
LPO-3	Function Naming Error	Minor	Fixed
LPO-4	Code Duplication	Informational	Fixed
MHE-1	Overflow Handling Error	Medium	Fixed
TRE-1	Permission Management Confusion	Major	Fixed
TRE-2	Out of Index	Minor	Fixed

3 Participant Process

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

Admin

- `withdraw_coins_from_treasury` : Allows a fee admin to withdraw coins from the treasury.
- `set_stable_pool_amp_factor_internal` : Set amplification factor for a stable pool.
- `set_swap_fee_multipliers_internal` : Set swap fee multipliers for traders.
- `execute_limit_order_order` : Executes a limit order using Atmos router.
- `update_config` : Updates global pump configuration.
- `set_swap_fee_protocol_allocation_bps` : Sets the swap fee protocol allocation in basis points.
- `toggle_pool_operations` : Toggles the pause state of pool operations, should halt all pool operations when active.
- `set_role` : Assign a role to an address.
- `remove_role` : Remove a role from an address.
- `execute_dca_order` : Executes a pending DCA order using Atmos router.

User

- `create_pool_stable` : Create a new stable pool.
- `create_pool_weighted` : Create a new weighted pool.
- `add_liquidity_stable` : Add liquidity to a stable pool.
- `add_liquidity_weighted` : Add liquidity to a weighted pool.
- `remove_liquidity` : Remove liquidity from a pool.
- `swap_exact_in_stable` : Perform exact input swap in stable pool.
- `swap_exact_in_weighted` : Perform exact input swap in weighted pool.
- `swap_exact_out_stable` : Perform exact output swap in stable pool.

- `swap_exact_out_weighted` : Perform exact output swap in weighted pool.
- `pool_balances_with_ref` : Get pool balances using an existing pool reference.
- `stable_pool_exists` : Check if a stable pool exists with given parameters.
- `create_limit_order_order` : Creates a new limit order with specified parameters.
- `cancel_order_entry` : Cancels an active limit order.
- `create<TokenType>` : Creates a new pump pool for token launch.
- `buy` : Executes token purchase from pump pool.
- `sell` : Executes token sale back to pump pool.
- `add_liquidity_stable_entry` : Adds liquidity to stable pool with safety checks.
- `add_liquidity_weighted_entry` : Adds liquidity to weighted pool with safety checks.
- `create_pool_stable_entry` : Creates new stable pool with initial liquidity.
- `create_pool_weighted_entry` : Creates new weighted pool with initial liquidity.
- `remove_liquidity_entry` : Remove liquidity from a pool.
- `set_stable_pool_amp_factor` : Set amplification factor for a stable pool.
- `set_swap_fee_multipliers` : Set swap fee multipliers for specific traders.
- `set_swap_fee_protocol_allocation_bps` : Set protocol allocation of swap fees.
- `swap_exact_in_stable_entry` : Perform exact input swap in stable pool.
- `swap_exact_in_weighted_entry` : Perform exact input swap in weighted pool.
- `swap_exact_out_stable_entry` : Perform exact output swap in stable pool.
- `swap_exact_out_weighted_entry` : Perform exact output swap in weighted pool.
- `swap_exact_in_multihop_entry<OutputToken>` : Perform multi-hop exact input swap.
- `create_dca_order` : Creates a new DCA order for automated periodic investments.
- `cancel_order_entry` : Cancels an active DCA order.
- `deposit_by_version<X>` : Handles token deposits with version compatibility.
- `withdraw_coin_as_fa<Coin>` : Withdraws coins with fungible asset conversion.

- `withdraw_coin_as_fa_and_deposit<Coin>` : Combines withdrawal and deposit operations.
- `withdraw_coins_as_fa<CoinU, CoinV, CoinW, CoinX, CoinY, CoinZ>` : Handles multi-coin withdrawals with conversion.
- `add_liquidity_stable<T0, T1, T2, T3, T4, T5>` : Adds liquidity to a stable pool by converting and depositing multiple coins.
- `add_liquidity_weighted<T0, T1, T2, T3>` : Adds liquidity to a weighted pool by converting and depositing multiple coins.
- `create_pool_stable<T0, T1, T2, T3, T4, T5>` : Creates a new stable pool with the specified parameters and initial liquidity.
- `create_pool_weighted<T0, T1, T2, T3>` : Creates a new weighted pool with the specified parameters and initial liquidity.
- `swap_exact_in_stable<T0>` : Performs a stable swap with exact input amount.
- `swap_exact_in_weighted<T0>` : Performs a weighted swap with exact input amount.
- `swap_exact_out_stable<T0>` : Performs a stable swap with exact output amount.
- `swap_exact_out_weighted<T0>` : Performs a weighted swap with exact output amount.

4 Findings

CUT-1 Incorrect Check in `deposit_by_version()`

Severity: Major

Status: Fixed

Code Location:

sources/coin_utils.move#68

Descriptions:

In the `deposit_by_version()`, the check for `paired_metadata` is incorrect.

```
  assert!(paired_metadata != fungible_asset::metadata_from_asset(&token),
EBOTH_TOKENS_MUST_BE_SAME);
```

Suggestion:

It is recommended to modify the code as follows to fix this issue.

```
  assert!(paired_metadata == fungible_asset::metadata_from_asset(&token),
EBOTH_TOKENS_MUST_BE_SAME);
```

Resolution:

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

DCA-1 Bounds Check Error

Severity: Medium

Status: Fixed

Code Location:

sources/dca.move#270-340

Descriptions:

In `execute_dca_order`, when counter = total_orders, `counter+1` will exceed `total_orders`.

Suggestion:

The condition `counter <= total_orders` should be changed to `counter < total_orders`.

Resolution:

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

DCA-2 Single Failure Will Block Subsequent Operations

Severity: Medium

Status: Acknowledged

Code Location:

sources/dca.move#270-340

Descriptions:

In `execute_dca_order`, if one execution fails (e.g., due to timeout or unsatisfactory swap results), all subsequent operations become blocked.

Suggestion:

It is recommended that if the execution fails, modify the execution time.

DCA-3 Stored Fungible Assets are Withdrawn by Users

Severity: Medium

Status: Acknowledged

Code Location:

sources/dca.move#178-237

Descriptions:

The stored fungible assets (FA) could be withdrawn by users, which may cause order execution failures.

Suggestion:

It is recommended that money should be locked in the account to prevent users from taking it out at will.

ENT-1 `pool_types` Value Error

Severity: Medium

Status: Fixed

Code Location:

sources/entry.move#723-839

Descriptions:

The `pool_types` corresponding to different pools in `swap_exact_in_multihop_entry()` should use different indexes, but this is all `*vector::borrow(&pool_types, 0)`

Suggestion:

It is recommended that different `pools` take corresponding `pool_types`.

Resolution:

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

GCO-1 Code Logic Flaws

Severity: Minor

Status: Fixed

Code Location:

sources/global_config.move#104-115

Descriptions:

In the method `set_swap_fee_protocol_allocation_bps()`

```
assert!(has_role(signer::address_of(manager), DEX_ADMIN_ROLE), ERR_UNAUTHORIZED);
assert!(exists<GlobalConfig>(resource_account::get_address()), ERR_INITIALIZED);
```

The order is reversed.

Suggestion:

It is recommended to swap the order.

Resolution:

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

GCO-2 Code Readability Issues

Severity: Informational

Status: Fixed

Code Location:

sources/global_config.move#215-227

Descriptions:

Use `is_initialized()` instead of `assert!(exists<GlobalConfig>(resource_account::get_address()), ERR_INITIALIZED);` to improve the readability.

Suggestion:

It is recommended to use `is_initialized()` instead of `assert!(exists<GlobalConfig>(resource_account::get_address()), ERR_INITIALIZED);` .

Resolution:

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

GCO-3 Unused Constant

Severity: Informational

Status: Fixed

Code Location:

sources/global_config.move#50;

sources/launchpad.move#50;

sources/liquidity_pool.move#117

Descriptions:

There is an unused constant in the contract.

```
const FEE_ADMIN_ROLE: u8 = 1;
```

Suggestion:

It is recommended to remove the unused constant if there's no further design.

Resolution:

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

LAU-1 Faulty Sell Function Logic Enables Free SUPRA Acquisition

Severity: Critical

Status: Fixed

Code Location:

sources/launchpad.move#511

Descriptions:

In the `sell` function, the correct logic should be to deduct the tokens from the user's account to exchange for the corresponding amount of `SUPRA` tokens. However, the current implementation of the contract does not deduct the tokens from the user's holdings. Instead, it takes out an amount of project tokens equivalent to `token_amount` from the pool itself (i.e., `pool_signer`) for the swap. As a result, the user does not actually pay any tokens but still receives `SUPRA` tokens, essentially allowing them to "free-ride" on the pool's funds. This flawed logic could quickly deplete the protocol's assets, leading to significant economic losses and security risks.

```
let (tokens_returned, supra_out_coins) = swap(
    pool_address,
    primary_fungible_store::withdraw(&pool_signer, pool.token, token_amount),
    fungible_asset::zero(option::extract(&mut supra_fa_metadata)),
    0,
    supra_out
);
```

Suggestion:

It is recommended to revise the `sell` function to ensure that the appropriate number of tokens is deducted from the user's account before the swap, rather than taking tokens from the pool account.

Resolution:

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

LAU-2 The Input Parameters are Incorrect

Severity: Major

Status: Fixed

Code Location:

sources/launchpad.move#248-284

Descriptions:

```
 pool.virtual_token_reserves = pool.virtual_token_reserves - tokens_out_amount;
        pool.virtual_supra_reserves = pool.virtual_supra_reserves - supra_out_amount;
        pool.virtual_token_reserves = pool.virtual_token_reserves +
fungible_asset::amount(&tokens_in);
        pool.virtual_supra_reserves = pool.virtual_supra_reserves +
fungible_asset::amount(&supra_in);

        // Verify pool value increased or stayed same
        assert_lp_value_is_increased_or_not_changed(
            pool.virtual_token_reserves,
            pool.virtual_supra_reserves,
            pool.virtual_token_reserves,
            pool.virtual_supra_reserves
        );
```

The two tokens passed in here are calculated, and none are passed before calculation, so `assert_lp_value_is_increased_or_not_changed()` can always pass, and it can also pass when the product of the two tokens decreases.

Suggestion:

It is recommended that two temporary variables be set to record the value of the token before calculation and then passed into the function.

Resolution:

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

LAU-3 No Refund Logic

Severity: Medium

Status: Fixed

Code Location:

sources/launchpad.move#419-489

Descriptions:

If the calculated `supra_required` of the `buy()` function is less than `max_supra_in`, the entire amount will be invested and no refund will be made.

Suggestion:

It is recommanded that the remaining amount of `max_supra_in - (supra_required + platform_fee)` be returned to the buyer.

Resolution:

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

LAU-4 Duplicate Platform Fee Charge in the `buy` Function

Severity: Medium

Status: Fixed

Code Location:

sources/launchpad.move#420

Descriptions:

In the `atmos_pump` module, the `buy` function withdraws (`supra_required + platform_fee`) amount of `supra` tokens from the user's account and passes them into the swap function, which already includes the platform fee. However, after the swap logic is executed, the protocol charges the platform fee once more from the user, resulting in a duplicate fee being applied. This causes users to be overcharged.

```
    let supra_in = coin::withdraw<SupraCoin>(buyer, supra_required + platform_fee);
    let supra_in_fa = coin::coin_to_fungible_asset<SupraCoin>(supra_in);
    // Execute swap
    let (tokens_out, supra_change) = swap(
        pool_address,
        in_zero,
        supra_in_fa,
        tokens_to_buy,
        0
    );

    // Take platform fee
    supra_account::deposit_coins<SupraCoin>(
        resource_account::get_address(),
        coin::withdraw<SupraCoin>(buyer, platform_fee)
    );
```

And in the `buy` function, the protocol mistakenly includes the platform fee as part of the input to the `swap` function and also adds the platform fee into the variable `pool.virtual_supra_reserves` for updates, but fails to deduct this fee from the pool. This causes the pool's `SUPRA` reserves to be artificially inflated, leading to distorted price calculations. As a result, in subsequent trades, users are required to provide more `SUPRA` tokens than necessary, and the pool's `K` value (constant product) becomes inaccurate. This flaw renders the assertion `assert_lp_value_is_increased_or_not_changed` ineffective, making it easy to bypass and posing a significant risk to the pricing and liquidity stability of the system.

```
let supra_in = coin::withdraw<SupraCoin>(buyer, supra_required + platform_fee);
        let supra_in_fa = coin::coin_to_fungible_asset<SupraCoin>(supra_in);
        // Execute swap
        let (tokens_out, supra_change) = swap(
            pool_address,
            in_zero,
            supra_in_fa,
            tokens_to_buy,
            0
        );

...
 pool.virtual_supra_reserves = pool.virtual_supra_reserves +
fungible_asset::amount(&supra_in);
```

Suggestion:

It is recommended to modify the `buy` function to pass only the actual token amount needed for purchase (`supra_required`) to the swap function, thereby avoiding the duplication of the platform fee.

Resolution:

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

LAU-5 Accuracy Issues

Severity: Minor

Status: Fixed

Code Location:

sources/launchpad.move#438-441

Descriptions:

```
let supra_required = (((pool.virtual_supra_reserves as u128) *
        (tokens_to_buy as u128) /
        ((pool.virtual_token_reserves - tokens_to_buy) as u128)) as u64) + 1;
```

If it is divisible here, then adding 1 will cause problems.

Suggestion:

It is recommanded that you should use the rounding up function

Resolution:

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

LOR-1 Multiple Execution Issues

Severity: Medium

Status: Fixed

Code Location:

sources/limit_orders.move#240-298

Descriptions:

`execute_limit_order_order` does not check `is_executed` to prevent multiple executions.

Suggestion:

It is recommended that adding a check for `is_executed`.

Resolution:

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

LPO-1 `fee_bps` Validation Error

Severity: Medium

Status: Fixed

Code Location:

sources/liquidity_pool.move#1961-1967

Descriptions:

`validate_swap_fee()` fee_bps 0.1% corresponds to 10, which is wrong here,but the code is 5

```
fun validate_swap_fee(fee_bps: u64) : bool {
      // Valid fee values: 0.01%, 0.05%, 0.3%, 1%
          fee_bps == 1 ||    // 0.01%
          fee_bps == 5 ||    // 0.05%
          fee_bps == 5 ||    // 0.1%
          fee_bps == 30 ||   // 0.3%
          fee_bps == 100     // 1%
   }
```

Suggestion:

It is recommended that the `fee_bps` corresponding to 0.1% be changed to 10

Resolution:

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

LPO-2 Loss of Precision

Severity: Minor

Status: Fixed

Code Location:

sources/liquidity_pool.move#1358-1375

Descriptions:

`compute_fees_given_amount_in_post_fee` should round up when calculating `total_amount_in` .

Suggestion:

`compute_fees_given_amount_in_post_fee` should round up when calculating `total_amount_in` .

Resolution:

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

LPO-3 Function Naming Error

Severity: Minor

Status: Fixed

Code Location:

sources/liquidity_pool.move#1096-1123

Descriptions:

`set_swap_fee_multipliers_internal` is named internal but is public.

Suggestion:

It is recommended that you change the function name.

Resolution:

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

LPO-4 Code Duplication

Severity: Informational

Status: Fixed

Code Location:

sources/liquidity_pool.move#1245-1260

Descriptions:

```
assert!(bps_demominator != 0, error::invalid_argument(EBPS_DENOMINATOR_ZERO));

        let total_fee_amount = (((swap_fee_bps as u128) * (amount_in as u128) /
(bps_demominator as u128)) as u64);
        let protocol_fee_bps = global_config::protocol_fee_ratio();

        assert!(bps_demominator != 0, error::invalid_argument(EBPS_DENOMINATOR_ZERO));
```

`assert` check for duplicates.

Suggestion:

It is recommended that you remove the second `assert`.

Resolution:

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

MHE-1 Overflow Handling Error

Severity: Medium

Status: Fixed

Code Location:

Math/sources/math_helper.move#15-33

Descriptions:

The `wrap_add()` function correctly handles overflow `value - (max_u128 - increment) - 1`, and the `wrap_sub()` function correctly handles overflow `max_u128 - (decrement - value) + 1`.

Suggestion:

It is recommanded that correctly modify overflow function calculation.

Resolution:

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

TRE-1 Permission Management Confusion

Severity: Major

Status: Fixed

Code Location:

sources/treasury.move#43

Descriptions:

Both `ROLE_FEE_ADMIN` and `ROLE_ORDER_EXECUTOR` have a value of 1, which may lead to permission confusion.

Suggestion:

It is recommended to encapsulate the method and use global functional functions to manage the permissions.

Resolution:

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

TRE-2 Out of Index

Severity: Minor

Status: Fixed

Code Location:

sources/treasury.move#167-180

Descriptions:

```
smart_table::add(&mut atmos_treasury.pool_treasury_details, pool, PoolTreasuryDetails
{
assets_fee_inflow: vector[0,0,0,0,0,0]
});
```

In the `add_fee_to_treasury` function, in the initialization pool, the asset type is 6, but `idx_asset` can be from 0 to 6, which is 7, and may exceed the index.

Suggestion:

It is recommended that `idx_asset<7` be changed to `idx_asset<6`.

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