Magma Finance

Audit Report

contact@bitslab.xyz

https://twitter.com/movebit_

Mon Jan 06 2025

Magma Finance Audit Report

1 Executive Summary

1.1 Project Information

Description	Magma is a decentralized exchange (DEX) built on the SUI blockchain, implementing Uniswap v3's concentrated liquidity model with ve3,3 gauge system for liquidity mining incentives.
Type	DEX
Auditors	MoveBit
Timeline	Mon Dec 23 2024 - Mon Jan 06 2025
Languages	Move
Platform	Others
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/MagmaFinanceIO/magma-core
Commits	4c4b1f4e5f547d21313fa432464a149448208533 2d4e7675916a057996d3d1fa1591b035389f2ec0 b6537ec6aac559f1cb51a55c3aef0f1d8be2f7c0 5f8a6e585c5590079ef332dd89eb0cb1755e98ee

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MOV	clmm/Move.toml	95fea5e05d52375189209f69a853fa33c1fa58b7
ACL	clmm/sources/acl.move	2bd944eb7aa41268b98869d9cd4c8d88c8380890
CMA	clmm/sources/clmm_math.move	de8c2034bf111f33a88138973ee417bd101f5a00
FAC	clmm/sources/factory.move	6a402b7aa1e6ebd916f72bcff927356b20db6bc0
POS	clmm/sources/position.move	8af94e6cc124d5156b503b6088a08203d61d47d4
REW	clmm/sources/rewarder.move	6eabf851ec7f644c476f14cff40b8f0b0610cc0e
PAR	clmm/sources/partner.move	848a43de6fa92fb141f79b3f074877d075672330
CON	clmm/sources/config.move	9dfad617e600b782cd2685c7c2eec789fe8555fe
POO	clmm/sources/pool.move	6cfdc591643e60210ada1a3ecb39dc77e9f3be68
UTI	clmm/sources/utils.move	d6b59eafdc6de5602e6a5ac019c95792353dba45
TIC	clmm/sources/tick.move	55e9fa90dad6866dba340101f03ed999e85f508f

TMA	clmm/sources/tick_math.move	6f176fc1c25c04621b02a69bd2d5366dad71928c
MOV1	caps/Move.toml	6b1134beadf256526ff114e0c33a9e0c08a923d5
GCA	caps/sources/gauge_cap.move	4413ed241131f0f5b62be835f42e248b39cd068f
MOV2	distribution/Move.toml	21eeb209affcfc50a23ce12c1296d8ecdb930514
VDA	distribution/sources/voting_dao.move	5f4b92a7a5613eeb6d8efda02997679750ec4af3
MTO	distribution/sources/magma_token.move	be622d1c8070a5ec4a53f32444efe7d44f28ca11
VRE	distribution/sources/voting_reward.move	da39a3ee5e6b4b0d3255bfef95601890afd80709
WTP	distribution/sources/proofs/whitelisted_token_pair.move	200605aebe1aa3cfb5a9b8739df69b0b3ce2f556
LOW	distribution/sources/proofs/lock_owner.move	7e5ded57a99655c21ffc7c589e9e40d8b983e4e1
GTF	distribution/sources/proofs/gauge_to_fee.move	e43de824ba6647bb341e2696ca5b6f264964f8d3
VOT	distribution/sources/voter.move	5fc00d9cd56bf77db17a1704fe45f259ca46574f
LMR	distribution/sources/rewards/locked_managed_reward.move	6d748a9b2e0bd31ca1b0ac64ff376551ef55ade0
FVR	distribution/sources/rewards/fee_voting_reward.move	85721ae3e76b5350164d8eb78c992e5e9453f372

REW1	distribution/sources/rewards/reward.move	a2ce28b4f31a484ee5fb33029165a404755d0838
FMR	distribution/sources/rewards/free_managed_reward.move	0c995f7600f2095a059149076305c318026d3927
BVR	distribution/sources/rewards/bribe_voting_reward.move	f756ee7756702db2e1ac1608cf040b759260cc75
RDC	distribution/sources/caps/reward_distributor_cap.move	0fea4c85e8082d26fcadab56599d6c6450992249
VCA	distribution/sources/caps/voter_cap.move	ae991ff0d5055fd724462d615260390a14e39313
NRC	distribution/sources/caps/notify_reward_cap.move	00c24aef85b7e4825a8cd19e1b5ef1130ad36aa8
RAC	distribution/sources/caps/reward_authorized_cap.move	3fa201e5d03b07fc879d1c9bfe8dc14d167d8516
TCA	distribution/sources/caps/team_cap.move	8801b3eb3884fd65b1d6018fc0a6b44001dae328
ECO	distribution/sources/emergency_council.move	9626ecc66fba6be8106cdbf8edf9d5171a33a353
COM	distribution/sources/common.move	a4c7d65dd64919086a63dcffabf3e9c9ad5c1a30
GAU	distribution/sources/gauge.move	efbeb7028dd0356352ebf3900a98442359608da6
RDI	distribution/sources/reward_distributor.move	c101bd81372b1f2788ae4220a93fd56b947fd129
VES	distribution/sources/voting_escrow.move	f3a830990d690ebbe0dbdfa92ddeaa8c23ac4e95

MIN	distribution/sources/minter.move	87761f244d15e086301a9fcc98d30fbd5392972d
ACL	sources/acl.move	4a6ca37f917f306f152d785365b3e5bdd2460691
CMA	sources/clmm_math.move	be0dee82f9c49dc6fdf33542a7fe922218f2bc4f
FAC	sources/factory.move	70de5aacbaaa2c2966e5a0141ffe0c3c2a7b7344
POS	sources/position.move	0bf79918b94f7c9fb93b14d564e9c9a0ae397107
REW	sources/rewarder.move	117ca98c5549672cda3892d5f778064ef0f1d47c
PAR	sources/partner.move	ee11e3dc6ca7b79c802f1afb4f572306bfb966e9
CON	sources/config.move	6007a19b9ea8cb7066c7627fc5582890b82a460c
POO	sources/pool.move	4b86a7999f686aa3fa3a57c72592b5946a1c1c56
UTI	sources/utils.move	14e11083d9864a490b987f66f7d6cc1946eabf1c
TIC	sources/tick.move	29b35ed9da2a51e459eba747e10c69b24d45e7fd
TMA	sources/tick_math.move	0317be8f83ec23e5bcff76d7da5ef7191d1b4b45

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	11	7	4
Informational	1	1	0
Minor	6	2	4
Medium	1	1	0
Major	3	3	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 Magma to identify any potential issues and vulnerabilities in the source code of the Magma Finance 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 11 issues of varying severity, listed below.

ID	Title	Severity	Status
ACL-1	The Role Scope Is Too Large And Exceeds The Number of Permission Types	Informational	Fixed
CMA-1	Spelling Error	Minor	Fixed
FAC-1	Gauge Preemptive Registration Causes Gauge-related Functions to Be Unavailable	Major	Fixed
GAU-1	Missing Reward Claim Verification Before Withdrawal	Medium	Fixed
MIN-1	Excessive Growth Rate Leading to Exponential Emissions	Major	Fixed
MIN-2	Excessive Decay Rate Leading to Near-Zero Emissions	Major	Fixed
PAR-1	Update Time Should Not Check Start Time Range	Minor	Acknowledged
POO-1	Pause Without Check The Current Status	Minor	Fixed

POO-2	Unused Constants	Minor	Acknowledged
RDC-1	Unsynchronized State Between Admin Action and Minter Object	Minor	Acknowledged
RDI-1	Update Failure Due to Period Limitation in Token Distribution	Minor	Acknowledged

3 Participant Process

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

Overview

Magma DEX combines the capital efficiency of concentrated liquidity with vote-escrowed tokenomics to create an advanced DeFi protocol. The key features include:

- Concentrated liquidity pools with customizable price ranges
- ve3,3 gauge system for liquidity mining incentives
- Protocol fee collection and distribution
- Partner referral system
- Advanced tick-based price management
- Secure position management and tracking

Core Components

Pool

The core AMM implementation that handles:

- Swap execution with concentrated liquidity
- Position management
- Fee collection and distribution
- Price range and tick management
- Protocol fee handling
- Gauge reward distribution

Position

Manages liquidity positions including:

- Position creation and modification

- Fee collection
- Reward tracking
- Position staking for gauge rewards

Reward System

The ve3,3 style reward system includes:

- Gauge voting and reward distribution
- Position staking/unstaking
- Reward accrual and claiming
- Fee sharing for staked positions
- Multiple reward token support
- Time-based reward emissions

Access Control

Role-based access control system for:

- Protocol fee management
- Pool parameter updates
- Partner management
- Reward distribution control

Key Features

Concentrated Liquidity

- Custom price range selection for LPs
- Improved capital efficiency
- Tick-based price management

Reward Distribution

- Vote-escrowed token governance
- Gauge voting for reward allocation
- Staking rewards for LPs
- Fee sharing for staked positions
- Flexible reward token support

Partner System

- Referral fee collection
- Customizable fee rates
- Partner reward tracking

Protocol Fees

- Configurable protocol fee rates
- Fee collection and distribution
- Protocol revenue sharing

Technical Details

Architecture

- Modular design with clear separation of concerns
- Extensive use of generics for token handling
- Comprehensive event emission for tracking
- Robust access control system

Security Features

- Role-based access control

- Input validation and bounds checking
- Safe arithmetic operations
- Pause functionality for emergencies

Math Implementation

- Precise tick math calculations
- Safe liquidity and fee computations
- Accurate reward distribution math

Usage

The protocol can be interacted with through:

- Direct smart contract calls
- Integration with frontend interfaces
- SDK implementations

4 Findings

ACL-1 The Role Scope Is Too Large And Exceeds The Number of Permission Types

Severity: Informational

Status: Fixed

Code Location:

sources/acl.move#23,56,67

Descriptions:

```
assert!(role < 128, ErrInvalidRole);
```

The number of permissions is limited to 128 or less, but there are only 5 permissions currently. You can consider narrowing the scope of the check to ensure the validity of the permission settings.

Suggestion:

It is recommended to narrow the scope of the check to the number of permissions you need.

Resolution:

The client replied:it is for leaving some space for future design.

CMA-1 Spelling Error

Severity: Minor

Status: Fixed

Code Location:

sources/clmm_math.move#24;

sources/gauge.move#419

Descriptions:

The following variable names are misspelled:

1. `current_liqidity` in `compute_swap_step()` should be spelled to `current_liquidity`.
2. `notifiy_reward` in `notifiy_reward()` should be spelled in `notify reward`.

Suggestion:

It is recommended to fix these typos.

Resolution:

The customer took our advice and fixed the issue.

FAC-1 Gauge Preemptive Registration Causes Gauge-related Functions to Be Unavailable

Severity: Major

Status: Fixed

Code Location:

sources/factory.move#94

Descriptions:

One `gauge` object corresponds to one `pool` object, which is determined by calling the option library function during initialization. In `pool.move`:

```
public(package) fun init_magma_distribution_gauger<A, B>(pool: &mut Pool<A, B>, gauger_id: ID) {
    pool.magma_distribution_gauger_id.fill(gauger_id)
}
```

In `option.move`:

```
public fun fill<Element>(t: &mut Option<Element>, e: Element) {
    let vec_ref = &mut t.vec;
    if (vec_ref.is_empty()) vec_ref.push_back(e)
    else abort EOPTION_IS_SET
}
```

The above settings result in the `Pool` only being able to set the `gauge` once. In `factory.move`

```
public fun return_new_gauger<A, B, C>(pool: &mut Pool<A, B>, ctx: &mut TxContext): Gauge<A, B, C> {//@audit anyone can call this
    //...
}
```

Since the association between `pool` creation and gauge is not atomic, an attacker can use the permissions of this function to create a `gauge` for the pool and send the returned `gauge` object to an address controlled by the attacker, which will cause the `gauge` to be unable to be passed as a parameter to other users.

Suggestion:

It is recommended to set the visibility of function `return_new_gauger()` to private function.

Resolution:

Gauge creation has been moved into Voter. Further more, the `return_new_gauge` fun is now only public(package) accessible.So it is impossible to pre-register the gauger.

GAU-1 Missing Reward Claim Verification Before Withdrawal

Severity: Medium

Status: Fixed

Code Location:

distribution/sources/gauge.move#358

Descriptions:

The function `withdraw_position` allows users to withdraw their staked position from the `Gauge`. However, it does not verify whether the user has claimed their pending rewards before allowing withdrawal. If a user withdraws without claiming their accumulated rewards, they may lose their entitled rewards, leading to potential dissatisfaction and financial loss.

Suggestion:

Before executing the withdrawal, add a check to ensure that the user has claimed all their rewards. This can be done by verifying the reward status from the staking record associated with `position_id`.

Resolution:

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

MIN-1 Excessive Growth Rate Leading to Exponential Emissions

Severity: Major

Status: Fixed

Code Location:

distribution/sources/minter.move#346

Descriptions:

The `epoch_grow_rate` is set to 103%, which results in the next_emissions being 203% of the `current_emissions`. This causes emissions to grow exponentially with each epoch. Over time, this unsustainable increase may lead to extreme inflation, destabilizing the system and depleting resources rapidly.

Suggestion:

It is recommended to adjust the `epoch_grow_rate` to a realistic percentage, such as 3% (300 in basis points).

Resolution:

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

MIN-2 Excessive Decay Rate Leading to Near-Zero Emissions

Severity: Major

Status: Fixed

Code Location:

distribution/sources/minter.move#350

Descriptions:

The `epoch_decay_rate` is set to **99%**, which results in the `next_emissions` shrinking to **1%** of the `current_emissions` in each epoch. This aggressive decay drastically reduces emissions, rendering the system ineffective and prematurely exhausting the rewards mechanism.

Suggestion:

It is recommended to adjust the `epoch_decay_rate` to a more reasonable value, such as 1% (100 in basis points), ensuring a gradual and controlled decrease in emissions.

Resolution:

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

PAR-1 Update Time Should Not Check Start Time Range

Severity: Minor

Status: Acknowledged

Code Location:

sources/partner.move#211

Descriptions:

The setting time should satisfy the following relationship: `end_time>start_time>now()` .But in codes it only satisfied: `end_time>start_time` and `end_time>now()` .This may cause the time setting to be incorrect.

```
public fun update_time_range(
    cfg: &config::GlobalConfig,
    partner: &mut Partner,
    start_time: u64,
    end_time: u64,
    clock: &clock::Clock,
    ctx: &mut TxContext
) {
    assert!(end_time > start_time, ErrInvalidEndTime);//@audit check start time >= now
    assert!(end_time > clock.timestamp_ms() / 1000, ErrInvalidEndTime);
    //...
}
```

Suggestion:

It is recommended to limit the setting of `start_time` :

```
assert!(start_time> clock.timestamp_ms() / 1000, ErrInvalidStartTime);
```

Resolution:

The client replied that it fits their design and start time in the past is totally allowed.

POO-1 Pause Without Check The Current Status

Severity: Minor

Status: Fixed

Code Location:

sources/pool.move#845;

sources/pool.move#1140

Descriptions:

When calling `pause()` and `unpause()` to change the state, there is no check whether the current state of the contract is already in the `paused` state or not.

```
public fun pause<CoinTypeA, CoinTypeB>(cfg: &config::GlobalConfig, pool: &mut Pool<CoinTypeA, CoinTypeB>, ctx: &mut TxContext) {//@audit check the current status is not paused
    cfg.checked_package_version();
    cfg.check_pool_manager_role(ctx.sender());
    pool.is_pause = true;
}
```

Suggestion:

It is recommended to add an extra check in `pause()` and `unpause()` . e.g. In `pause()` :

```
public fun pause<CoinTypeA, CoinTypeB>(cfg: &config::GlobalConfig, pool: &mut Pool<CoinTypeA, CoinTypeB>, ctx: &mut TxContext) {//@audit check the current status is not paused
    cfg.checked_package_version();
    assert!(!pool.is_pause, ErrPoolPaused);//@audit add this check
    cfg.check_pool_manager_role(ctx.sender());
    pool.is_pause = true;
}
```

Resolution:

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

POO-2 Unused Constants

Severity: Minor

Status: Acknowledged

Code Location:

sources/pool.move#26,28,34,39

Descriptions:

There are multiple unused constants in the contract. Below are some examples:

```
const ErrLiquidityUnderflow: u64 = 2; //@audit unused constant
const ErrNotEnoughLiquidity: u64 = 4;//@audit unused constant
const ErrInvalidFixedCoinType: u64 = 10;//@audit unused constant
const ErrInvalidProtocolFeeRate: u64 = 15;//@audit unused constant
```

Suggestion:

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

Resolution:

The client replied that it fits their design and it is for future developing.

RDC-1 Unsynchronized State Between Admin Action and Minter Object

Severity: Minor

Status: Acknowledged

Code Location:

distribution/sources/caps/reward_distributor_cap.move#69

Descriptions:

The `start` function allows the admin to update the `RewardDistributor`'s `start_time`, `last_token_time`, and `minter_active_period` independently. If the admin calls this function without updating the associated `Minter` object, it can lead to desynchronization between the `RewardDistributor` and the `Minter` states. This inconsistency may cause unpredictable behavior, such as incorrect reward calculations or distribution logic failures.

Suggestion:

It is recommend to make this function private if calling it drectly is not in need and make sure it fits your design.

Resolution:

The client replied:By design, the reward distributor cap will be transferred to the `Minter`. This makes sure that the `Minter` and the `RewardDistributor` have a one-one relationship.

RDI-1 Update Failure Due to Period Limitation in Token Distribution

Severity: Minor

Status: Acknowledged

Code Location:

distribution/sources/reward_distributor.move#102;

distribution/sources/voting_escrow.move#722,1236

Descriptions:

The loop within the token distribution logic restricts updates to a maximum of 20 periods (i < 20) or 255 (i < 255). If the required number of periods to update exceeds 20 or 255, the update process will fail to complete, leaving certain periods unprocessed. This limitation can result in incomplete token distribution, potentially impacting the contract's functionality and the fairness of the reward distribution.

Suggestion:

It is recommended to remove the hardcoded limit of fixed iterations and allow the loop to process all required periods dynamically to ensure all updates are completed.

Resolution:

The client replied that it serves as a means to save gas consumption. Since SUI does not care that much about gas, it might be reasonable to remove this. But 255 means no updates in almost 5 years. I think it's more than enough to cover normal situations. This limitation does not usually cause any impact and can basically handle most scenarios.

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.