Magma Finance

Audit Report

contact@bitslab.xyz

https://twitter.com/movebit_

Thu Oct 16 2025

Magma Finance Audit Report

1 Executive Summary

1.1 Project Information

Description	Magma focuses on building a sustainable liquidity incentive engine, aligning the interest of traders, governance participants, and liquidity providers, aiming to deliver a DEX of the best user experience and capital efficiency
Type	DEX
Auditors	MoveBit
Timeline	Mon Jun 09 2025 - Thu Oct 16 2025
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/MagmaFinanceIO/magma-core
Commits	1a0d32d9934e45f1e37e01787118eb3fc17e23f5 8469b5b8c67ca66000fd2c3bfab248099037ec8e e318966d39f95f5fa3072eb8a3101508b1af9b53

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MOV5	almm/Move.toml	08e53f1c7250274a4f40e888e64a060c65bc593e
BTR	almm/sources/bin_tree.move	3e4ae2ca94f5bd7c89d88d9e079995ab0299b590
REW2	almm/sources/rewarder.move	f297120f18227afb681a417f0136d840c3f3837c
PRI	almm/sources/price.move	a6f8edbcfff1a4e5143b2f4bf4913fd7677cc88d
PAI	almm/sources/pair.move	0e0d267bf7bf2965e70164d2d7cfc1b9b51097e5
FAC1	almm/sources/factory.move	58130250cf4cdffdad486ddcea06c6d765889ae4
POS1	almm/sources/position.move	e7771a5ef96fbb017f27e1059863c0ccb2639a7f
CON2	almm/sources/constants.move	e05415d3ee446c8193442c540bd07497f7875161
FEE	almm/sources/fee.move	584e3c755b91c2630615c16a0cd768c2dad3c446
PIN	almm/sources/position_info.move	dc93fe634d7fbfeb043eea288fdc2f4f5939de3a

BIN	almm/sources/bin.move	a7d5a0c443a1b14f185a8942a2ac841dbd510df3

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	15	15	0
Informational	1	1	0
Minor	7	7	0
Medium	5	5	0
Major	2	2	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 15 issues of varying severity, listed below.

ID	Title	Severity	Status
FAC-1	`revoke_protocol_fee_cap` Permission Was Not Correctly Revoked	Major	Fixed
FAC-2	`unstaked_liquidity_fee_rate` Lacks Setter Function After Initialization in `factory.init()`	Medium	Fixed
FAC-3	`revoke_admin` Verification Of The Object Is Insufficient	Minor	Fixed
FAC-4	`set_max_bins_in_position` Missing Permission Verification	Minor	Fixed
FAC-5	`update_preset` and `add_preset` Lack Of Scope Check	Minor	Fixed
REW-1	`emergent_withdraw` Incorrect Calling Sequence	Minor	Fixed
BIN1-1	`get_shares_and_effective_amounts_in` Calculation Code Error	Medium	Fixed

BIN1-2	`stake_liquidity` Equal Functions Should Be Internal Functions	Minor	Fixed
BTR1-1	`contains` Parameter Range Check Is Missing	Medium	Fixed
PAI1-1	`burn` Can Never Be Invoked	Major	Fixed
PAI1-2	Unchangeable `pause` Status in Prevents Use of Pause Mechanism	Medium	Fixed
PAI1-3	Missing Pair Pause State Check in `mint()` Function	Medium	Fixed
PAI1-4	`update_references` Missing Range Verification	Minor	Fixed
PAI1-5	`mint_update_bin_internal` Fee Calculation May Overflow	Minor	Fixed
POS1-1	`ErrNotEnougthSharesToDecrease` Rename To `ErrNotEnoughSharesToDecrease`	Informational	Fixed

3 Participant Process

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

Admin

- `open_bin_step_preset` : Open an existing preset
- `close_bin_step_preset` : Close an open preset
- `add_preset` : Add a new preset for the specific combination of base_fee and bin_step
- `remove_preset` : Remove an existing preset
- `update_preset` : Update an existing preset parameter
- `set_composition_enabled` Enable or disable the combined cost function
- `set_static_fee_parameters_external` : Set the static fee parameters for the trading pair
- `force_decay` : Forcibly attenuate the reference variable for volatility
- `set_protocol_variable_share` : Set the proportion of the agreement's share in variable fees
- `emergent_withdraw` : Urgently withdraw the specified quantity of reward tokens from the global reward vault
- `set_version` Updated version
- `set_paused` Set the pause state
- `set_unstaked_liquidity_fee_rate` : Set an unstaked liquidity fee rate

User

- `swap_with_partner` : Execute token exchanges and obtain rates from partner objects to charge additional fees
- `settle_rewarders` : Settle and update the status of all rewards

- `swap` : Conduct token exchange operations
- `mint_by_amounts` : Based on the precise number of tokens provided by the user, add liquidity to multiple specified bins and create a new liquidity position
- `mint` : Based on the total number of tokens provided by the user and their percentage distribution in each bin, add liquidity and create a new liquidity position
- `burn` : Remove a specified number of liquidity shares from a specified bin from an existing liquidity position
- `burn_position` : Remove all liquidity from a liquid position and destroy the position object
- `raise_position_by_amounts` : Increase liquidity to an existing liquidity position by an exact amount
- `shrink_position` : Reduce the liquidity of all bins in an existing liquidity position by a certain proportion
- `collect_fees` : Withdraw the earned transaction fees for the specified liquidity position
- `collect_reward` : Withdraw the specific type of rewards already earned for the specified liquidity position

Publisher

- `grant_admin` : Create a new administrator credential and grant it to the specified receiver address
- `revoke_admin` : Remove an AdminCap from the allowed_admin list
- `set_default_partner` : Set or update a default Partner ID
- `grant_protocol_fee_cap` : Create a protocol fee collection voucher and grant it to the specified receiver address
- `revoke_protocol_fee_cap` : Remove a ProtocolFeeCap from the allowed_protocol_fee_cap list
- `update_factory_id` : Update the factory contract ID associated with the trading pair

ProtocolFeeCap

- `collect_protocol_fees` : Extract the accumulated agreement fees in the trading pair

RewarderManager

- `update_rewarder_emission` : Update the reward release rate of the specified rewards

GaugeCap

- `update_magma_distribution_growth_global` : Update the global growth value of the staking reward
- `stake_in_magma_distribution` : Stake a liquid position in the reward pool to earn rewards
- `unstake_from_magma_distribution` : Unstaked one liquid position from the reward pool
- `sync_magma_distribution_reward` : Reward rate reserve quantity and end time of the synchronous staking reward pool
- `collect_magma_distribution_gauge_fees` : Withdraw the accumulated fees from the staking reward pool
- `grow_distribution_growth` : Update the allocation reward growth of the specified bin_id and return the growth difference

4 Findings

FAC-1 `revoke_protocol_fee_cap` Permission Was Not Correctly Revoked

Severity: Major

Status: Fixed

Code Location:

almm/sources/factory.move#242

Descriptions:

The function wrongly removed the cap from the factory.allowed_admin collection. In fact, it should be removed from the factory.allowed_protocol_fee_cap collection. This causes the function to attempt to remove the cap from a completely unrelated collection (the list of administrator credentials allowed_admin). If the cap happens to also exist in allowed_admin, then an administrator permission will be accidentally revoked. If cap does not exist in allowed_admin, the remove operation will fail silently (i.e., do nothing).

```
public fun revoke_protocol_fee_cap(factory: &mut Factory, _: &Publisher, cap: ID) {
    assert!(factory.allowed_protocol_fee_cap.contains(&cap));
    factory.allowed_admin.remove(&cap);
    emit(EventRevokeProtocolFeeCap {
        cap,
    });
}
```

Suggestion:

Modify to:

```
public fun revoke_protocol_fee_cap(factory: &mut Factory, _: &Publisher, cap: ID) {
    assert!(factory.allowed_protocol_fee_cap.contains(&cap));
    factory.allowed_protocol_fee_cap.remove(&cap);
    emit(EventRevokeProtocolFeeCap {
```

```
        cap,
    });
}
```

Resolution:

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

FAC-2 `unstaked_liquidity_fee_rate` Lacks Setter Function After Initialization in `factory.init()`

Severity: Medium

Status: Fixed

Code Location:

almm/sources/factory.move#122

Descriptions:

In the `factory.init()` function, the protocol assigns a default value of `0` to `unstaked_liquidity_fee_rate` during contract deployment. However, there is no subsequent function to update this value.

```
fun init(otw: ALMM_FACTORY, ctx: &mut TxContext) {
    let publisher = package::claim(otw, ctx);
    let mut factory = Factory {
        id: object::new(ctx),
        pairs: table::new(ctx),
        pairs_iter: vector::empty(),
        allowed_admin: vec_set::empty(),
        allowed_protocol_fee_cap: vec_set::empty(),

        presets: table::new(ctx),
        opened_bin_steps: vec_map::empty(),

        available_bin_steps: table::new(ctx),

        unstaked_liquidity_fee_rate: 0,

        default_partner: option::none(),

        enable_composition: DEFAULT_COMPOSITION_FEE_ENABLED,

        max_bins_in_position: DEFAULT_MAX_BINS_IN_POSITION,
    };
```

```
    let factory_id = object::id(&factory);
    factory.grant_admin(&publisher, ctx.sender(), ctx);
    transfer::share_object(factory);
    transfer::public_transfer(publisher, ctx.sender());

    emit(EventCreateFactory { factory_id });
}
```

Suggestion:

It is recommended to add a `set` function to allow updating the `unstaked_liquidity_fee_rate`.

Resolution:

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

FAC-3 revoke_admin Verification Of The Object Is Insufficient

Severity: Minor

Status: Fixed

Code Location:

almm/sources/factory.move#192

Descriptions:

The revoke_admin, revoke_protocol_fee_cap, update_factory_id function signature contains the _: &Publisher parameter. This indicates that the function's intention is to only allow the module's publisher to call it to undo the administrator. However, the code only checks whether the caller passed in a Publisher object, but does not verify which module this Publisher object belongs to. And other similar functions all uniformly use assert! Check (publisher.from_package()), and this check should also apply to these functions

```
public fun revoke_admin(factory: &mut Factory, _: &Publisher, admin_cap: ID) {
    assert!(factory.allowed_admin.contains(&admin_cap));
    factory.allowed_admin.remove(&admin_cap);
    emit(EventRevokeAdmin {
        cap: admin_cap
    });
}
```

Suggestion:

Add the corresponding checks

```
assert!(publisher.from_package<Factory>());
```

Resolution:

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

FAC-4 `set_max_bins_in_position` Missing Permission Verification

Severity: Minor

Status: Fixed

Code Location:

almm/sources/factory.move#232

Descriptions:

Although the parameters define the need for &AdminCap (administrator credentials), it is not used at all to verify the identity of the caller. And other similar functions all use factory.check_admin(cap); Verify

```
public fun set_max_bins_in_position(self: &mut Factory, _: &AdminCap, max: u64) {
    let old = self.max_bins_in_position;
    assert!(old != max);
    self.max_bins_in_position = max;
    emit(EventMaxBinsInPosition { old, new: max });
}
```

Suggestion:

Modify to

```
public fun set_max_bins_in_position(self: &mut Factory, cap: &AdminCap, max: u64) {
    factory.check_admin(cap);
    let old = self.max_bins_in_position;
    assert!(old != max);
    self.max_bins_in_position = max;
    emit(EventMaxBinsInPosition { old, new: max });
}
```

Resolution:

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

FAC-5 `update_preset` and `add_preset` Lack Of Scope Check

Severity: Minor

Status: Fixed

Code Location:

almm/sources/factory.move#303

Descriptions:

The functions `add_preset` and `update_preset` allow administrators to add or update parameter presets for dynamic rates, but do not verify the validity of the `protocol_share` (protocol share ratio) parameter. Due to the existence of

```
const MAX_PROTOCOL_SHARE: u16 = 2500; // 25%
```

parameter limits the maximum value of protocol_share, so the configuration value needs to be verified to ensure that its value is within the valid range

Suggestion:

Add checks

```
assert!(protocol_share <= constants::MAX_PROTOCOL_SHARE, ...);
```

Resolution:

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

REW-1 `emergent_withdraw` Incorrect Calling Sequence

Severity: Minor

Status: Fixed

Code Location:

almm/sources/rewarder.move#133

Descriptions:

Judging from the naming of the event structure field after_amount, its design intention is to record the remaining balance of reward tokens in the vault after the execution of this emergency withdrawal operation, but it is called before the vault. Youdaoplaceholder0 function What is recorded is the balance before the operation. Although it does not directly lead to financial loss, it will cause inaccurate data in the on-chain Event Log

```
public fun emergent_withdraw<RewardType>(_admin_cap: &config::AdminCap, cfg:
&config::GlobalConfig, vault: &mut RewarderGlobalVault, amount: u64):
Balance<RewardType> {
    cfg.checked_package_version();
    event::emit(EmergentWithdrawEvent{
        reward_type: type_name::get<RewardType>(),
        withdraw_amount: amount,
        after_amount: vault.balance_of<RewardType>(),
    });
    vault.withdraw_reward<RewardType>(amount)
}
```

Suggestion:

Modify to

```
public fun emergent_withdraw<RewardType>(_admin_cap: &config::AdminCap, cfg:
&config::GlobalConfig, vault: &mut RewarderGlobalVault, amount: u64):
Balance<RewardType> {
    cfg.checked_package_version();
```

```
    vault.withdraw_reward<RewardType>(amount)
    event::emit(EmergentWithdrawEvent{
        reward_type: type_name::get<RewardType>(),
        withdraw_amount: amount,
        after_amount: vault.balance_of<RewardType>(),
    });
}
```

Resolution:

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

BIN1-1 `get_shares_and_effective_amounts_in` Calculation Code Error

Severity: Medium

Status: Fixed

Code Location:

almm/sources/bin.move#232

Descriptions:

Liquidity is mathematically defined as the geometric mean of the quantities of the two tokens. Specifically, Liquidity = sqrt(amount_x * amount_y). The variable user_liquidity actually represents the product of amount_x * amount_y. Therefore, to obtain the LP share proportional to the liquidity from this product, it is necessary to take its square root. This implementation is linear (user_liquidity / 2), and it does not correctly convert the token product to liquidity units. This may lead to inaccurate calculation of the share of the first liquidity provider (LP) and may introduce deviations for subsequent transactions or liquidity operations. The / 2 in the code seems like an empirical value or a simplified approximation, lacking rigorous mathematical support.

```
public fun get_shares_and_effective_amounts_in(reserve_x: u64, reserve_y: u64,
amount_x: u64, amount_y: u64, price_q128: u256, total_supply: u64): (u64, u64, u64) {
    let user_liquidity = get_liquidity(amount_x, amount_y, price_q128);
    if (user_liquidity == 0) {
        return (0, 0, 0)
    };
    let bin_liquidity = get_liquidity(reserve_x, reserve_y, price_q128);
    if (bin_liquidity == 0 || total_supply == 0) {
        return (uint_safe::safe64((user_liquidity / 2) >> constants::scale_offset()), amount_x,
amount_y)
        // return (uint_safe::safe64(magma_math_u256::sqrt(user_liquidity) >>
(constants::scale_offset() / 2)), amount_x, amount_y)
    };
```

```
let shares = user_liquidity * (total_supply as u256) / bin_liquidity;
let effective_liquidity = shares * bin_liquidity / (total_supply as u256);
```

Suggestion:

Calculate using annotated code

Resolution:

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

BIN1-2 `stake_liquidity` Equal Functions Should Be Internal Functions

Severity: Minor

Status: Fixed

Code Location:

almm/sources/bin.move#197

Descriptions:

These functions should only be managed and called by the core logic within the protocol and should be modified to public(package).

```
public fun stake_liquidity(self: &mut Bin, liquidity: u256) {
    self.staked_liquidity = self.staked_liquidity + liquidity;
}

public fun unstake_liquidity(self: &mut Bin, liquidity: u256) {
    self.staked_liquidity = self.staked_liquidity - liquidity;
}

public fun grow_fee_x(self: &mut Bin, growth: u256) {
    self.fee_growth_x = self.fee_growth_x + growth;
}
public fun grow_fee_y(self: &mut Bin, growth: u256) {
    self.fee_growth_y = self.fee_growth_y + growth;
}
```

Suggestion:

Change to public(package)

Resolution:

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

BTR1-1 `contains` Parameter Range Check Is Missing

Severity: Medium

Status: Fixed

Code Location:

almm/sources/bin_tree.move#31

Descriptions:

The function accepts an id of type u32, but there is no assertion (assert!) within the function body. To ensure that the value of this id is within the valid range of 24 bits (i.e., less than 2^24).

```
public fun contains(self: &BinTree, id: u32): bool {
    let key2 = (id >> 8) as u256;
    if (!self.level2.contains(&key2)) {
        return false
    };
    let leaves = *self.level2.get(&key2);
    leaves & ((1 << ((id & (U8_MAX as u32)) as u8)) as u256) > 0
}
```

Suggestion:

Add parameter verification

```
assert!((id >> 24) == 0, ErrInvalidId);
```

Resolution:

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

PAI1-1 `burn` Can Never Be Invoked

Severity: Major

Status: Fixed

Code Location:

almm/sources/pair.move#1124

Descriptions:

There is an assertion assert! In the code. It will check whether the conditions are true. If they are false, the transaction will be immediately terminated and all status changes will be rolled back. Because the condition here is always false, any attempt to call the burn function will fail immediately

Suggestion:

If the deletion function is not allowed to be called, if it needs to be called, an assertion must be removed

Resolution:

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

PAI1-2 Unchangeable `pause` Status in Prevents Use of Pause Mechanism

Severity: Medium

Status: Fixed

Code Location:

almm/sources/pair.move#426

Descriptions:

When Pair is initialized, the protocol sets the pair's `pause` status to `false`.

Multiple parts of the protocol check this `pause` status to determine if operations should be halted.

```
assert!(!self.pause, ErrPaused);
```

However, there is no function provided later in the code to update or modify this `pause` status, meaning the pause mechanism cannot be utilized once the contract is deployed.

Suggestion:

It is recommended to implement a function with proper access control to update the `pause` status.

Resolution:

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

PAI1-3 Missing Pair Pause State Check in `mint()` Function

Severity: Medium

Status: Fixed

Code Location:

almm/sources/pair.move#1070-1074

Descriptions:

In the `mint()` function, the protocol interacts with the pair but does not verify whether the pair is in a paused state.

Suggestion:

It is recommended to add a check to verify whether the associated pair is paused.

Resolution:

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

PAI1-4 update_references Missing Range Verification

Severity: Minor

Status: Fixed

Code Location:

almm/sources/pair.move#1845

Descriptions:

In blockchain and any system with temporal dependence, time should flow unidirectionally and monotonically increase. self.last_update_timestamp records the time point of the last status update, while current_timestamp is the time point of this update. Logically, current_timestamp must be greater than or equal to self.last_update_timestamp, so this test can be added to ensure the mathematical correctness of internal time-related calculations (such as volatility decay, dynamic cost adjustment)

Suggestion:

Add checks

```
ssert!(timestamp >= self.time_of_last_update, ....);
```

Resolution:

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

PAI1-5 `mint_update_bin_internal` Fee Calculation May Overflow

Severity: Minor

Status: Fixed

Code Location:

almm/sources/pair.move

Descriptions:

The core of the vulnerability lies in the multiplication operation of fee_x * (self.params.protocol_share as u64).

In a Move, u64 type a maximum of 18446744073709551615 (2 ^ 64-1)

Both fee_x and self.params.protocol_share are treated as u64 types. When the value of fee_x (transaction fee) is very large, the product of it and protocol_share (protocol share ratio) is very likely to exceed the maximum upper limit of u64.

```
let protocol_fee_x_ = if (fee_x > 0) {
    fee_x * (self.params.protocol_share as u64) / (constants::basis_point_max() as u64)
} else { 0 };
```

Suggestion:

Use integer types with a higher bit width (such as u128) to perform intermediate calculations

Resolution:

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

POS1-1 `ErrNotEnougthSharesToDecrease` Rename To `ErrNotEnoughSharesToDecrease`

Severity: Informational

Status: Fixed

Code Location:

almm/sources/position.move#5

Descriptions:

There are spelling mistakes in the words , `ErrNotEnougthSharesToDecrease` Rename To `ErrNotEnoughSharesToDecrease`

Suggestion:

Correct the wrong spelling

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.