Audit Report

Margined Power

v1.0

September 26, 2023

Table of Contents

License

Disclaimer

THE CONTENT OF THIS AUDIT REPORT IS PROVIDED “AS IS”, WITHOUT REPRESENTATIONS AND WARRANTIES OF ANY KIND.

THE AUTHOR AND HIS EMPLOYER DISCLAIM ANY LIABILITY FOR DAMAGE ARISING OUT OF, OR IN CONNECTION WITH, THIS AUDIT REPORT.

THIS AUDIT REPORT IS ADDRESSED EXCLUSIVELY TO THE CLIENT. THE AUTHOR AND HIS EMPLOYER UNDERTAKE NO LIABILITY OR RESPONSIBILITY TOWARDS THE CLIENT OR THIRD PARTIES.

This audit has been performed by

Oak Security

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of This Report

Oak Security has been engaged by Osmosis Grants Company to perform a security audit of Margined Protocol's Power smart contracts.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the following target:

Repository	https://github.com/margined-protocol/power
Commit	`4648abf616660017d72fcbfd7ba6106c7384feee`
Scope	All contracts except `contracts/mocks` were in scope.

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line-by-line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
4. Report preparation

Functionality Overview

Margined Power is a decentralized power perpetual protocol and multi-chain margin engine for CosmWasm networks. The key innovation of Margined Power is the use of concentrated liquidity pools to enable price discovery of the contracts.

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending,** **Acknowledged**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than in a security audit and vice versa.

Code Quality Criteria

The auditor team assesses the codebase's code quality criteria as follows:

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	Medium-High	Readability is in line with other CosmWasm projects in the ecosystem.
Level of documentation	Low-Medium	While a high-level overview documentation is available, there is little inline documentation and no technical specification, for example concerning the core calculation logic, design decisions, and intended behaviors of parameters.
Test coverage	Medium-High	`cargo tarpaulin` reports a test coverage of `28.08%`, `422/1503` lines covered. Note that this does not include integration tests that are implemented using `osmosis-test-tube` testing framework.

Summary of Findings

No	Description	Severity	Status
1	Vault safeness incorrectly calculated due to the usage of wrong decimals	Critical	Resolved
2	The `power` contract no longer works if `apply_funding_rate` is not called for 48 hours	Critical	Resolved
3	User funds are lost when using `Burn` to withdraw collateral	Critical	Resolved
4	Power contract can be used if not yet opened or if paused	Major	Resolved
5	The staking contract does not handle transfers of staked coins properly	Major	Resolved
6	Normalization factor could be manipulated	Minor	Acknowledged
7	Unpause delay too small to react	Minor	Resolved
8	Missing `fee_rate` validation	Minor	Resolved
9	Lack of denomination and pool validation	Minor	Resolved
10	Contracts should implement a two-step ownership transfer	Minor	Resolved
11	`get_index` does not return a scaled index	Minor	Resolved
12	Events/attributes are not always emitted	Informational	Resolved
13	Inefficient execution could be improved	Informational	Resolved
14	The power contract can be opened multiple times	Informational	Resolved
15	Lack of role-based access controls for the pausing mechanism	Informational	Acknowledged
16	Widespread usage of generic errors	Informational	Resolved
17	Token authority check can be bypassed	Informational	Resolved
18	Unused code	Informational	Resolved

Detailed Findings

1. Vault safeness incorrectly calculated due to the usage of wrong decimals

Severity: Critical

The `power` contract checks if each user vault is solvent as a pre-condition to perform the mint, burn, deposit, withdraw, and liquidate operations. However, the calculations done in the `get_status` function in `contracts/margined-power/src/vault.rs:319` could yield incorrect results.

When the collateral is turned into `decimal` type in line `344`, `power_decimals` is used instead of `base_decimals`, which represents the amount of decimals from the Power denomination instead of the one from the collateral. This error could cause insolvent vaults to be wrongly deemed "safe", allowing further operations or "safe" vaults to be considered "unsafe" and liquidatable.

Recommendation

We recommend using `base_denom` when converting the collateral into a `decimal` type.

Status: Resolved

2. The `power` contract no longer works if `apply_funding_rate` is not called for 48 hours

Severity: Critical

In `contracts/margined-power/src/handle.rs`, the `apply_funding_rate` function calls `calculate_normalisation_factor`, which checks if the last funding update `state.last_funding_update` has occurred in the last `MAX_TWAP_PERIOD`, which is set to 48 hours. If not, it returns an error.

While it is assumed that the `apply_funding_rate` function is regularly called by the foundation or community, a potential exists that it is not called for extended periods, for example due to network outages. If it is not called for 48 hours, most core functions, such as `handle_mint_power_perp`, `handle_burn_power_perp`, `handle_liquidation`, `handle_deposit`, and `handle_withdrawal` will be unusable given the fact that they all invoke `apply_funding_rate`.

Recommendation

We recommend using a fixed TWAP period and removing this check or introducing a recovery method that can resolve this issue.

Status: Resolved

3. User funds are lost when using `Burn` to withdraw collateral

Severity: Critical

The `power` contract's `handle_burn_power_perp` function requires the user to send both the power tokens to be burned and the collateral tokens to be withdrawn in `contracts/margined-power/src/handle.rs:263` and `264`. However, the additional collateral sent this way is never accounted for.

Both `amount_to_burn` and `amount_to_withdraw` are supplied as arguments to `VAULTS.burn`, which deducts each amount from the corresponding field in the storage. Finally, `amount_to_withdraw` is transferred to the user. The whole operation results in the user sending the requested amount of collateral tokens, getting that amount deducted from the vault but only receiving the amount reimbursed once instead of twice. This leads to lost the user losing funds.

Recommendation

We recommend following one of these two approaches:

- Reimburse `amount_to_withdraw` twice at the end instead of once.
- Redesign the function not to require the user to send any additional collateral but to provide an additional argument to be passed down at the entry point specifying the amount of collateral to be withdrawn.

Status: Resolved

4. Power contract can be used if not yet opened or if paused

Severity: Major

For the `power` contract to be usable, it must be opened explicitly by executing `SetOpen` message. It can also be paused and unpaused by executing `Pause` and `UnPause` messages. These actions modify the variable `state.is_open`. However, the code does not contain a check if the value of this variable is `true` before user actions are performed. Hence it is possible to interact with the contract, even when it has not been opened yet or is paused.

Recommendation

We recommend querying `state.is_open` before every action and returning an error when it is `false`.

Status: Resolved

5. The staking contract does not handle transfers of staked coins properly

Severity: Major

The function `handle_unstake` in `contracts/margined-staking/src/handle.rs:209` decrements `staked_amounts` of a user whenever they unstake. Since `staked_amounts` is not updated when the staked coin (`staked_denom`) is transferred, this calculation can underflow when transferred coins are later unstaked. For instance, when Alice transfers her staked coins to Bob (who does not have a staked position) and Bob tries to unstake, he will not be able to.

Recommendation

We recommend disabling transfers of staked coins or updating the `staked_amounts` variable when transfers happen.

Status: Resolved

6. Normalization factor could be manipulated

Severity: Minor

The `power` contract's `apply_funding_rate` function calculates the normalization factor in `contracts/margined-power/src/funding.rs:20` only if it has not been already calculated at the current timestamp (and hence in the current block). The `calculation_normalisation_factor` function queries the oracle in TWAP mode using the difference from the last update of `normalisation_factor` and the current timestamp as its period. This is the first step in the five major external functionalities: mint, burn, liquidate, deposit, and withdraw.

Given the mentioned restriction, performing consecutive calls in different blocks with just one second of difference between their timestamps could be possible, forcing the oracle to return information of just a one-second period. This mostly defeats the benefits of using the TWAP mechanism, as it is possible to briefly manipulate the Oracle in favor of the user before interacting with the contract.

Recommendation

We recommend setting a minimum period for TWAP queries.

Status: Acknowledged

The client acknowledges that there is some room for manipulation but due to the bounds placed on the index price in the calculation, adequate protection from oracle attacks already exists.

7. Unpause delay too small to react

Severity: Minor

The `power` contract's pause mechanism allows anonymous users to unpause the contract 24 hours after it has been paused. Although this feature follows best practices to reduce centralization concerns, the period of one day may be too small in case of an incident response situation or after the discovery of a critical vulnerability in the code.

Recommendation

We recommend increasing the period for anonymous users to unpause the contract, for example, to one week. Moreover, we recommend implementing a disaster recovery procedure such that users can still withdraw their funds when the contract is paused in response to a vulnerability.

Status: Resolved

8. Missing `fee_rate` validation

Severity: Minor

In `contracts/margined-power/src/contract.rs:60`, `config.fee_rate` is missing validation for the value to be between 0 to 1. Without validation, misconfiguration may go unnoticed and will eventually be difficult to correct.

Similarly, the `handle_update_config` in `contracts/margined-power/src/handle.rs:49` should have the same validation for the `config.fee_rate`.

Recommendation

We recommend adding `fee_rate` validation.

Status: Resolved

9. Lack of denomination and pool validation

Severity: Minor

In several instances in the `staking` and `power` contracts, denomination and pools lack validation:

- `deposit_denom` and `reward_denom` in `contracts/margined-staking/src/contract.rs:59` and `61`
- `power_denom`, `base_denom`, `power_pool`, and `base_pool` in `contracts/margined-power/src/state.rs:59-69`
- `token` in `contracts/margined-collector/src/state.rs:8`

For example, for the last instance, if an invalid denom is added to the list of allowed tokens, it will not be usable within the `collector` contract.

Further, if a typo is made when removing a token the same "token not added" error will be raised as when a valid but non-existent token is to be removed.

Recommendation

We recommend querying the token factory to check if a token denomination has a valid token associated. Likewise, we recommend querying pools to confirm whether the expected denominations align with those within that pool.

Status: Resolved

10. Contracts should implement a two-step ownership transfer

Severity: Minor

The contracts within the scope of this audit allow the current owner to execute a one-step ownership transfer. While this is common practice, it presents a risk for the ownership of the contract to become lost if the owner transfers ownership to the incorrect address. A two-step ownership transfer will allow the current owner to propose a new owner, and then the account that is proposed as the new owner may call a function that will allow them to claim ownership and actually execute the config update.

In addition, the `staking` contract does not have a mechanism to transfer ownership at all. This is problematic as in case of a suspected compromise, no action could be taken.

Recommendation

We recommend implementing a two-step ownership transfer. The flow can be as follows:

1. The current owner proposes a new owner address that is validated.
2. The new owner account claims ownership, which applies the configuration changes.

Status: Resolved

11. `get_index` does not return a scaled index

Severity: Minor

The `get_index` function in `contracts/margined-power/src/query.rs:78-95` does not scale the calculated index, as the naming suggests. Also, it does the same as the `get_unscaled_index` function, making one of these functions redundant.

We classify this issue as minor given that the `get_index` function is used as part of a query entry point that third-party contracts may rely on.

Recommendation

We recommend querying the scaled entry point instead.

Status: Resolved

12. Events/attributes are not always emitted

Severity: Informational

Multiple functions within the scope of this audit do not emit event attributes. It is best practice to emit attributes to improve the contracts' usability and support off-chain event listeners and blockchain indexers.

The following functions do not emit events or attributes:

- In `contracts/margined-collector/src/handle.rs:56`, the `send_token` function is lacking attributes, such as `amount` and `to_address`.
- In `contracts/margined-staking/src/handle.rs:89`, the `handle_claim` function is lacking attributes, such as `amount`.
- In `contracts/margined-staking/src/handle.rs:152`, the `handle_stake` function is lacking attributes, such as `staked_amounts`.
- In `contracts/margined-staking/src/handle.rs:209`, the `handle_unstake` function is lacking attributes, such as `sent_funds`.

Recommendation

We recommend emitting event attributes as described above.

Status: Resolved

13. Inefficient execution could be improved

Severity: Informational

The `update_rewards` function of the `staking` contract in `contracts/margined-staking/src/distributor.rs:36` lacks optimization of the execution path when `block_rewards` is zero: Although the result could be short-circuited if there are no `block_rewards`, the whole function is executed, consuming unnecessary resources and gas.

Recommendation

We recommend returning zero in line `40` if there are no `block_rewards`.

Status: Resolved

14. The `power` contract can be opened multiple times

Severity: Informational

In `contracts/margined-power/src/handle.rs:41` within the `handle_open_contract` function, it is not checked if the contract was already opened. An administrator can therefore open it multiple times successfully. While this has no negative security impact, it can confuse consumers because the event `open_contract` is emitted every time the function is executed.

Recommendation

We recommend disallowing opening an already opened contract.

Status: Resolved

15. Lack of role-based access controls for the pausing mechanism

Severity: Informational

The contracts within scope implement pausing mechanisms, which is in line with best practices. However, all of the administrative functions of the contract are centralized in the owner role, which goes against the principle of least privilege.

Segregating the pauser role has the additional benefit of swifter reactions in case of need when assigned to an EOA compared to the admin that may be a multi-sig or managed by a governance contract.

Recommendation

We recommend implementing a separate pauser role that can turn trigger the pausing mechanism.

Status: Acknowledged

16. Widespread usage of generic errors

Severity: Informational

The contracts within scope of this audit make widespread use of generic errors. Although not a security issue, generic errors are discouraged as they may decrease the maintainability of the codebase.

Recommendation

We recommend making use of custom errors.

Status: Resolved

17. Token authority check can be bypassed

Severity: Informational

The `power` contract implements the `SetOpen` entry point to unpause the contract for the first time after checking that the contract is the authority of the power token.

However, there is no mechanism to ensure that this entry point is used the first time – the authority check can be bypassed by directly calling `UnPause`.

Recommendation

We recommend checking the authority upon instantiation instead.

Status: Resolved

18. Unused code

Severity: Informational

Across the codebase, instances of unused errors, constants, fields within structs and functions exist in the following locations. Unused code should be removed as it can decrease the readability and maintainability of the codebase.

- `InsufficientFunds` in `contracts/margined-power/src/errors.rs:19`.
- `UserStakeNotFound` in `contracts/margined-staking/src/errors.rs:13`.
- `InvalidLiquidation` in `contracts/margined-staking/src/errors.rs:19`.
- `InsufficientFunds` in `contracts/margined-staking/src/errors.rs:22`.
- `VaultDoesNotExist` in `contracts/margined-staking/src/errors.rs:25`.
- `Stakers` constant in `contracts/margined-staking/src/state.rs:9`.
- `average_staked_amounts` field within the `UserStake` struct is never read and only assigned once in `contracts/margined-staking/src/distributor.rs:85`.
- `DECIMALS`, `DECIMAL_PLACES`, and `SCALE_FACTOR` constants in `contracts/margined-staking/src/contract.rs:25-28`.
- `quote_decimals` field in the `Config` struct in `contracts/margined-power/src/state.rs:31`.
- `DECIMAL_TWO` constant in `contracts/margined-power/src/helpers.rs:15`.
- `execute_transfer` function in `contracts/margined-collector/src/messages.rs:3-8`.
- `update` and `mint` functions in `contracts/margined-power/src/vault.rs:122` and `149` are redundant and one of them is unused.

Recommendation

We recommend removing unused code.

Status: Resolved