By
Quit

Contract Review
Issue date
09/4/2025

Overview

The following is a review of `Gridle` and `GridleToken`, two components of a betting system for a grid-based game where players place wagers on cells in a grid, each containing payout multipliers. Players bet on where a chart will intersect the grid, with winners receiving payouts based on the multiplier of their chosen cell.

Contracts in scope for this review include:

- `contracts/grid/Gridle.sol`
- `contracts/grid/GridleToken.sol`

This review is based on SHA `6d9fae402943cf25cf02ac24da5e3fac38fe8fb7`, and aims to identify security vulnerabilities, opportunities for gas optimization, and general best practice recommendations with regards to the contracts in scope. The review should not be considered an endorsement of the project, nor is it a guarantee of security.

Findings

I-01: Missing event emission for key state changes

Severity: Informational

`setSigner` and `setWithdrawAddress` in both contracts allows admins to update key state variables, but don't emit events to log these changes. Recommend adding events and emitting them in both functions.

I-02: Incorrect ordering of functions

Severity: Informational

Functions should be grouped according to their visibility and ordered:

- constructor
- receive function (if exists)
- fallback function (if exists)
- external
- public
- internal
- private

Within a grouping, place the view and pure functions last.
Source: https://docs.soliditylang.org/en/latest/style-guide.html#order-of-functions

I-03: Redundant reserved parameter in withdrawal functions

Severity: Informational

The `withdrawERC20` and `withdrawEth` functions require callers to specify a reserved amount to leave in the contract, but since the caller must query the current balance off-chain to calculate this parameter anyway, the same logic could determine the exact withdrawal amount directly. Recommend replacing the `reserved` parameter with a direct `amount` parameter to withdraw.

G-01: Use of error strings

Severity: Gas Optimization

Recommend using custom errors to revert, as strings in Solidity are very expensive. This will save a bit of gas at deploy, as well as each time a revert is hit. While some are present, there are still error strings used throughout the contracts as well.

L-01: Auto-withdrawal uses potentially stale `systemBalance`

Severity: Low

The automatic withdrawal mechanism triggers when `signId > lastSignId`, but this logic doesn't account for the delay between signature creation and transaction execution. The `systemBalance` parameter represents the backend's view of required reserves at the time the signature was generated, not when the deposit transaction actually executes. The impact is limited since administrators can manually adjust reserves, and the signature can only be as stale as the deadline allows, but it does exist. Recommend removing auto-withdrawal to simplify the system, or using very strict deadlines within signature payloads.

L-02: Refunds can compromise operational reserves

Severity: Low

The automatic withdrawal mechanism optimizes contract balance based on current `systemBalance` requirements, but subsequent refunds can reduce the balance without coordination between these systems. When a user deposits funds, `_autoWithdraw` calculates and withdraws excess funds. If an admin later issues a refund through `refundEth` or `refundToken`, the contract balance drops without re-evaluating reserves. Recommend adding reserve validation to refund functions to ensure `address(this).balance - refundAmount >= effectiveMinTarget` based on current `systemBalance`, implementing a mechanism to coordinate refunds with the auto-withdrawal system, or performing strict offchain checks before issuing refunds.

Summary

The `Gridle` and `GridleToken` contracts provide a solid foundation for the grid-based betting system. The core architecture is straightforward and correctly prevents replay attacks through order ID tracking. Most (or all) of the actual game logic exists offchain, leading to the usual centralization risks associated with protocols designed in this manner.

The automatic withdrawal feature adds significant complexity that may not justify its benefits. The coordination between auto-withdrawal optimization and manual refunds creates potential issues where refunds could leave insufficient reserves for active games. Additionally, the reliance on potentially stale `systemBalance` data in signatures could lead to issues as well, albeit easily resolved by using `topUp`. The system would likely be more robust and easier to operate by removing auto-withdrawal entirely and relying on manual balance management through the existing withdrawal functions.

The remaining issues are primarily code quality improvements like adding events for state changes and using custom errors for gas efficiency, none of which pose security risks but would improve the contracts' readability and efficiency.