LinumLabs

Flow - Audit

Prepared by Linum Labs AG

2024-11-22

linumlabs.com

Executive Summary

This audit covers the Flow decentralized roulette platform, which supports users placing straight bets on a virtual roulette table. The audit reviewed the core Roulette.sol smart contract.

Protocol Summary

Overview

This system is a decentralized Roulette game integrated with an ERC-4626 vault. Players can bet on Roulette outcomes using Flow's native currency (wrapped as WFlow) or native assets, with provably fair randomness provided by Flow's random beacon. The vault enables users to deposit assets, earn shares, and redeem them later, with a tax on profits benefiting the house. Risk management ensures the house remains solvent, limiting payouts to a percentage of available funds. Spins occur at defined block intervals, and outcomes are cached for efficient reward distribution.

Audit Scope

```
../contracts/Roulette.sol
```

Audit Results

Summary

Repository	https://github.com/wt-xyz/flow-roulette
Commit	4a1af0d...
Timeline	October 28th - November 1st

Issues Found

Bug Severity	Count
Critical	1
High	1
Medium	2
Low	1
Informational	3
Total Findings	8

Summary of Issues

Description	Severity	Status
totalRemainingPayouts is not scaled consistently with *MAX_PAYOUT_RATE*	Critical	Resolved
Loss of user winnings if spin outcome is not cached within next spin interval	High	Resolved
ERC4626 Vault Inflation	Medium	Resolved

Tax_Rate on withdrawal does not allow users to withdraw their complete assets	Medium	Resolved
Precision loss when placing side bets	Low	Acknowledged
Gambling compliance and responsible gambling practices	Informational	Acknowledged
Miner manipulation on Flows random number generator	Informational	Acknowledged
Use calldata instead of memory for function arguments that do not get mutated	Informational	Resolved

Issues

Critical Severity

1. *totalRemainingPayouts* is not scaled consistently with *MAX_PAYOUT_RATE*

Description: The *totalRemainingPayouts* state allows the contract to determine the payout which is yet to be distributed to the users for winning bets. The issue here is that the state is used to determine both the maximum possible payout for bets placed and the maximum payout amount for the winning bets. The state is scaled inconsistently with the *MAX_PAYOUT_RATE*.

Potential Risk: This could result in inconsistent *totalRemainingPayouts* and cause unexpected integer underflow **here**.

Suggested Mitigation: Consider maintaining multiple states for potential payouts for total bets placed and *totalRemainingPayouts* for winning bets, and scale *MAX_PAYOUT_RATE* consistently throughout the contract.

Flow: Fixed in the PR

Linum Labs: Verified

High Severity

2. **Loss of user winnings if spin outcome is not cached within next spin interval**

Description: In the Roulette contract the function *claimWinnings()* allows players to claim their winning if the player has placed a winning bet. The spin's outcome is cached using the function *cacheLastSpinOutcome()*, and the function *getLastResolvedSpin()* determines the last spin number. The issue here is if the function is not executed within the next spin interval then the function *cacheLastSpinOutcome()* would not be able to cache the outcome for the previous spins and the outcome of the spin would being UNDEFINED, resulting in the user not being able to withdraw their bets as there was no outcome for the spin. Furthermore, the unsettledBets state is prematurely reset when determining the spin outcome when executing the *cacheLastSpinOutcome()* function, even if previous bets are still pending.

Potential Risk: This would result in the user not being able to withdraw their bets as there was no outcome for the spin and inaccurate accounting of the state *unsettledBets*.

Suggested Mitigation: To ensure accurate accounting and prevent user funds from being locked, prioritize resolving the outcome of the previous spin ID where bets have been placed. Enhance the *cacheLastSpinOutcome()* function to accurately determine past spin outcomes, and reset the *unsettledBets* state only when the outcome of the previous spins are resolved.

Flow: Fixed in PR

Linum Labs: Verified

Medium Severity

3. ERC4626 Vault Share Inflation

Description: The first depositor of an *ERC4626* vault can maliciously manipulate the share price by depositing the lowest possible amount (1 wei) of liquidity and then artificially inflating *ERC4626.totalAssets*.

This can inflate the base share price as high as 1:1e18 early on, forcing all subsequent deposits to use this share price as a base. Worst case, due to rounding down, if this malicious initial deposit front-runs someone else's deposit, this depositor will receive 0 shares and lose his deposited assets.

Potential Risk: This could result in users receiving significantly fewer/negligible shares than expected for their deposits, potentially leading to substantial financial losses.

Suggested Mitigation: A small sacrificial deposit could be placed with the creation of the protocol to enforce the ratio of assets to shares, as seen in discussion here.

Alternatively, For the first deposit, mint a fixed amount of shares, e.g. 10**decimals()

```javascript
if (_totalAssets_ == 0) {
    shares = 10**decimals;
} else {
    shares = totalSupply * assets_ / _totalAssets_;
}
```

Flow: Fixed in PR

Linum Labs: Verified

4. **Tax_Rate on withdrawal does not allow users to withdraw their complete assets**

Description: The *maxWithdraw()* function calculates the maximum amount of the underlying asset that can be withdrawn from a user's balance in the Vault. However, it currently does not account for the *TAX_RATE* applied to profits. This oversight can result in users being unable to withdraw their full balance, leading to unexpected *INSUFFICIENT_SHARES()* reverts due to inaccurate calculations.

Potential Risk: This could result in unexpected revert when the user attempts to withdraw maximum withdrawable assets computed using the function *maxWithdraw()*.

Suggested Mitigation: Consider taking *TAX_RATE* into account when computing the maximum Withdrawal assets using the function *maxWithdraw()*.

Flow: Fixed in PR

Linum Labs: Verified

Low Severity

5. **Precision loss when placing side bets**

Description: When placing side bets, the total bet amount is divided among multiple outcomes. Due to potential integer division and rounding errors, there might be slight discrepancies in the final distribution of bets compared to the intended proportions.

Potential Risk: This would result in minor precision loss of around 20 wei on users winning bets.

Suggested Mitigation: Consider rounding up the final reward, to avoid the precision loss.

Flow: Acknowledged, Known issue.
Linum Labs: Acknowledged.

Informational Severity

6. Gambling compliance and responsible gambling practices

Description: To ensure responsible and transparent operations, gambling platforms must adhere to the following:

Record Keeping: Maintain detailed records of all user transactions, identity verifications, and suspicious activity reports for a legally mandated period.
Responsible Gambling Measures: Implement robust measures to promote responsible gambling practices, including:

- Self-Exclusion: Allow users to self-exclude from gambling activities for specified durations.
- Deposit Limits: Enable users to set personalized deposit limits.
- Reality Checks: Provide regular reminders to users about their gambling activity.
- Time-Outs: Offer options for users to take temporary breaks.

Potential Risk: **Non-compliance with regulatory requirements:** Failure to adhere to strict gambling regulations, including record-keeping, KYC/AML, and responsible gambling measures, could result in severe penalties, fines, and reputational damage.
Suggested Mitigation:
Robust Compliance Framework: Implement a comprehensive compliance program aligning with local and international regulations.

Advanced Record-Keeping: Utilize secure systems to maintain detailed records of all user transactions and activities.

Effective Responsible Gambling Measures: Offer tools like self-exclusion, deposit limits, reality checks, and time-outs.

Continuous Monitoring: Regularly monitor user behavior and implement automated systems to flag suspicious activity.
Flow: Acknowledged.
Linum Labs: Acknowledged.

7. Miner manipulation on Flows random number generator

Description: The *Roulette* contract uses Flow's random number generator to introduce an element of chance into the roulette spin. While miner manipulation of the random number might be possible, this aspect is beyond the scope of the current audit.

Potential Risk: If miners could potentially manipulate the random number generation process, this could lead to unfair outcomes and undermine user trust.
Suggested Mitigation:
Flow: Acknowledged.
Linum Labs: Acknowledged.

8. Use calldata instead of memory for function arguments that do not get mutated

Description: Mark data types as `calldata` instead of `memory` where possible. This makes it so that the data is not automatically loaded into memory. If the data passed into the function does not need to be changed (like updating values in an array), it can be passed in as `calldata`. The one exception to this is if the argument must later be passed into another function that takes an argument that specifies `memory` storage.

Potential Risk: This would lead to increased gas usage than required.
Suggested Mitigation: To optimize gas usage in your Solidity functions, mark data types as `calldata` instead of `memory` wherever applicable. This prevents unnecessary data loading into memory. Use `calldata` for function arguments that do not require changes within the function, except when passing them into another function that explicitly requires `memory` storage.
Flow: Fixed in PR .
Linum Labs: Verified

Disclaimer

This report is based on the materials and documentation provided to Linum Labs Auditing for the purpose of conducting a security review, as outlined in the Executive Summary and Files in Scope sections. It's important to note that the results presented in this report may not cover all vulnerabilities. Linum Labs Auditing provides this review and report on an as-is, where-is, and as-available basis. By accessing and/or using this report, along with associated services, products, protocols, platforms, content, and materials, you agree to do so at your own risk. Linum Labs Auditing disclaims any liability associated with this report, its content, and any related services and products, to the fullest extent permitted by law. This includes implied warranties of merchantability, fitness for a particular purpose, and non-infringement. Linum Labs Auditing does not warrant, endorse, guarantee, or assume responsibility for any third-party products or services advertised or offered through this report, its content, or related services and products. Users should exercise caution and use their best judgment when engaging with third-party providers. It's important to clarify that this report, its content, access, and/or usage thereof, including associated services or materials, should not be considered or relied upon as financial, investment, tax, legal, regulatory, or any other form of advice.