OpenBlock

EigenLayer Rewards v2 Audit Report

OpenBlock Labs

Contents

1. Executive Summary

OpenBlock Labs performed an in-depth audit of EigenLayer's Rewards V2 implementation, focusing primarily on the critical components of the rewards calculation logic and the integrations of both Sidecar V1 and V2. The audit highlighted key features

introduced in the Rewards V2 update, including operator-directed rewards and the accuracy of snapshots for take rate (operator split) adjustments.

Audit Details

Audit Scope	Rewards v2 & Sidecar Implementation
Repository	https://github.com/Layr-Labs/sidecar
Date	November 25th, 2024 - December 20th, 2024

Risk Level Classification

Risk Level	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

Action Required for risk levels

Critical: Must be fixed immediately, especially if already deployed to production.

High: Must be addressed before deployment, or as soon as possible if not yet deployed.

Medium: Should be fixed to improve functionality or reduce potential risks.

Low: Could be resolved if time permits, as it has minimal impact.

Findings Count

Risk Level	Amount
Critical	2
High	0
Medium	0
Low	3

Summary of Key Findings and Resolutions

Title	Risk Level	Status
End_time Calculation Error in Snapshots	Critical	Fixed
Incorrect Handling of Take Rate Changes	Critical	Fixed
Lack of Refund Mechanism for Unallocated Staker Incentives	Low	Acknowledged
Operator-Directed Reward Calculation Relies on Smart Contract Criteria	Low	Acknowledged
Minor Typos in Rewards Files	Low	Fixed

2. Fixes and Acknowledgments

EigenLayer has resolved all critical issues identified in the audit, including the end_time calculation error, the incorrect handling of take rate changes, and minor typos in file names. These fixes have been thoroughly audited by OpenBlock Labs.

Low-risk findings, such as reliance on smart contract criteria and the lack of refundability for unallocated incentives, have been acknowledged by EigenLayer.

3. Issues Identified

1. end_time Calculation Error in Snapshots

1.1 Risk Level: Critical

1.2 Status: Fixed

1.3 Root Cause

The issue lies in how the `end_time` is being calculated in the `operator_avs_split_windows` CTE. Specifically, **1 day is subtracted** from the `end_time` when constructing time windows.

```
operator_avs_split_windows as (
 SELECT
        operator, avs, split, snapshot_time as start_time,
        CASE
                -- If the range does not have the end, use the current timestamp truncated to 0 UTC
                WHEN LEAD(snapshot_time) OVER (PARTITION BY operator, avs ORDER BY snapshot_time) is null THEN date_trunc('day', TIMESTAMP '{{.cutoffDat

                -- need to subtract 1 day from the end time since generate_series will be inclusive below.
                ELSE LEAD(snapshot_time) OVER (PARTITION BY operator ORDER BY snapshot_time) - interval '1 day'
                END AS end_time
 FROM active_operator_splits
),
```

These time windows (consisting of `start_time` and `end_time`) are then **exploded** into daily snapshots using the `generate_series` function.

Due to the subtraction of one day from `end_time`, the last day of a time window is excluded when generating daily snapshots. This results in missing data for specific days when the operator changes the take rate more than once.

```sql
-- Generate a snapshot for each day in the range
final_results as (
    SELECT
        operator,
        avs,
        split,
        d AS snapshot
    FROM
        cleaned_records
            CROSS JOIN
        generate_series(DATE(start_time), DATE(end_time) - interval '1' day, interval '1' day) AS d
)
select * from final_results
```

1.4 Impact

- Leap Days in Data: If an operator changes its take rate multiple times, the exclusion of the last day (because of the `-1` logic) causes missing snapshots, leading to incomplete daily data.
- This results in gaps in the generated snapshots, affecting metrics that rely on accurate daily time-series data.

1.5 Reproducible Scenario

Consider the following example:

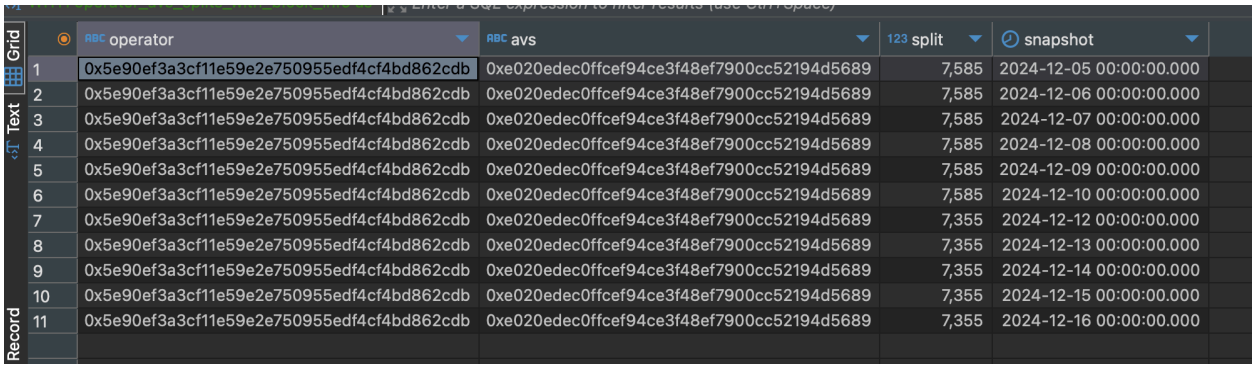
1. Operator: `0x5e90ef3a3cf11e59e2e750955edf4cf4bd862cdb`
2. AVS: `0xe020edec0ffcef94ce3f48ef7900cc52194d5689`
3. Take Rate Changes:
    - First Take Rate: `7585` activated on `2024-12-05`
    - Second Take Rate: `7355` activated on `2024-12-12`
4. Windows Generated by `operator_avs_split_windows`:

Take Rate	Start Time	End Time
7585	2024-12-05	2024-12-11
7355	2024-12-12	2024-12-17

When `generate_series` explodes these windows into daily snapshots:

- **Expected Rows**: 12 days (from `2024-12-05` to `2024-12-17`).
- **Actual Rows**: 11 days because `2024-12-11` is excluded from the first time window (7585).

This exclusion occurs because `end_time` is truncated by one day during the construction of windows.



	operator	avs	split	snapshot
1	0x5e90ef3a3cf11e59e2e750955edf4cf4bd862cdb	0xe020edec0ffcef94ce3f48ef7900cc52194d5689	7,585	2024-12-05 00:00:00.000
2	0x5e90ef3a3cf11e59e2e750955edf4cf4bd862cdb	0xe020edec0ffcef94ce3f48ef7900cc52194d5689	7,585	2024-12-06 00:00:00.000
3	0x5e90ef3a3cf11e59e2e750955edf4cf4bd862cdb	0xe020edec0ffcef94ce3f48ef7900cc52194d5689	7,585	2024-12-07 00:00:00.000
4	0x5e90ef3a3cf11e59e2e750955edf4cf4bd862cdb	0xe020edec0ffcef94ce3f48ef7900cc52194d5689	7,585	2024-12-08 00:00:00.000
5	0x5e90ef3a3cf11e59e2e750955edf4cf4bd862cdb	0xe020edec0ffcef94ce3f48ef7900cc52194d5689	7,585	2024-12-09 00:00:00.000
6	0x5e90ef3a3cf11e59e2e750955edf4cf4bd862cdb	0xe020edec0ffcef94ce3f48ef7900cc52194d5689	7,585	2024-12-10 00:00:00.000
7	0x5e90ef3a3cf11e59e2e750955edf4cf4bd862cdb	0xe020edec0ffcef94ce3f48ef7900cc52194d5689	7,355	2024-12-12 00:00:00.000
8	0x5e90ef3a3cf11e59e2e750955edf4cf4bd862cdb	0xe020edec0ffcef94ce3f48ef7900cc52194d5689	7,355	2024-12-13 00:00:00.000
9	0x5e90ef3a3cf11e59e2e750955edf4cf4bd862cdb	0xe020edec0ffcef94ce3f48ef7900cc52194d5689	7,355	2024-12-14 00:00:00.000
10	0x5e90ef3a3cf11e59e2e750955edf4cf4bd862cdb	0xe020edec0ffcef94ce3f48ef7900cc52194d5689	7,355	2024-12-15 00:00:00.000
11	0x5e90ef3a3cf11e59e2e750955edf4cf4bd862cdb	0xe020edec0ffcef94ce3f48ef7900cc52194d5689	7,355	2024-12-16 00:00:00.000

1.6 Recommended Fix

1. **Avoid subtracting 1 day** when setting the `end_time` in `operator_avs_split_windows` and `operator_pi_split_windows`. Instead, use the interval that fully captures the last day.

```
-- Get the range for each operator, avs pairing
operator_avs_split_windows as (
 SELECT
     operator, avs, split, snapshot_time as start_time,
     CASE
         -- If the range does not have the end, use the current timestamp truncated to 0 UTC
         WHEN LEAD(snapshot_time) OVER (PARTITION BY operator, avs ORDER BY snapshot_time) is null THEN date_trunc('day', cast(now() as timestamp))
         ELSE LEAD(snapshot_time) OVER (PARTITION BY operator ORDER BY snapshot_time)
         END AS end_time
 FROM active_operator_splits
```

2. Incorrect Handling of Take Rate Changes

2.1 Risk Level: Critical

2.2 Status: Fixed

2.3 Root Cause

The issue arises in the logic of the `decorated_operator_avs_splits` CTE, which determines whether a take rate change is active or not by identifying overlapping changes for the same operator and AVS. Specifically:

1. Take Rate Change Delay Logic:
    - When an operator sets a new take rate for an AVS, there is a **7-day delay period** before the change becomes active.
    - If an operator sets multiple take rate changes **within this 7-day delay period**, the delay resets for each new change, and the operator must wait another full 7 days for the latest change to take effect.
2. Current Logic Issue:

    In the following code snippet:

```
decorated_operator_avs_splits as (
    select
        rops.*,
        -- if there is a row, we have found another split that overlaps the current split
        -- meaning the current split should be discarded
        case when rops2.block_time is not null then false else true end as active
    from ranked_operator_avs_split_records as rops
    left join ranked_operator_avs_split_records as rops2 on (
        rops.operator = rops2.operator
                and rops.avs = rops2.avs
        -- rn is orderd by block and log_index, so this should encapsulate rops2 occurring afer rops
        and rops.rn > rops2.rn
        -- only find the next split that overlaps with the current one
        and rops2.rounded_activated_at <= rops.rounded_activated_at
    )
),
```

The `rounded_activated_at` column from both `rops` and `rops2` is being compared. This causes a **flawed overlap check** because:

- `rops2.block_time` (the timestamp of the take rate change) will always have a value except the initial take rate change transaction.

- As a result, the `CASE` statement evaluating `rops2.block_time is not null` will **always be true** for all rows except the initial take rate transaction.
- Consequently, **all subsequent take rate changes** are marked as `inactive` (i.e., `active = false`), which is incorrect.

2.4 Impact

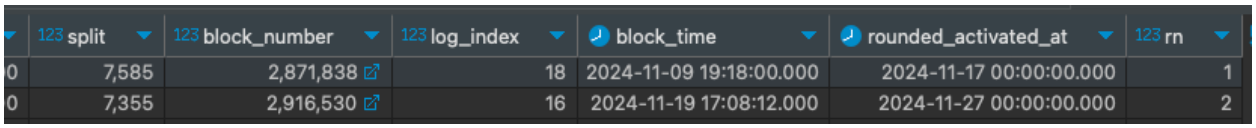
1. Incorrect Take Rate Status:
    - All take rate changes after the initial change are incorrectly flagged as `inactive`.
    - This prevents the pipeline from accurately identifying which take rate is valid at any given time.
2. Business Logic Violation:
    - The intended 7-day delay reset logic is not implemented correctly, leading to errors in determining when a take rate change should become active.
    - Metrics relying on the pipeline, such as operator earnings, will be inaccurate.

2.5 Reproducible Scenario

Assume the following take rate changes for an operator:

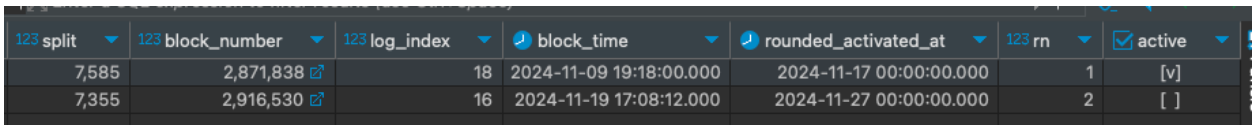
1. First Take Rate Change:
    - Take Rate: `7585` (Take Rate - 1)
    - Block Time: `2024-11-09 19:18:00.000`
    - Rounded Activated At: `2024-11-17 00:00:00.000`
2. Second Take Rate Change:
    - Take Rate: `7355` (Take Rate - 2)
    - Block Time: `2024-11-19 17:08:12.000` (10 days later)
    - Rounded Activated At: `2024-11-27 00:00:00.000`

Since the second take rate change occurs **after the 7-day delay period** of the first take rate change, **both take rates should be valid.**



split	block_number	log_index	block_time	rounded_activated_at	rn
0	7,585	2,871,838	18	2024-11-09 19:18:00.000	2024-11-17 00:00:00.000	1
0	7,355	2,916,530	16	2024-11-19 17:08:12.000	2024-11-27 00:00:00.000	2

When this data is passed into the `decorated_operator_avs_splits` CTE, the logic incorrectly detects the **second take rate change as overlapping** and sets the `active` column to `false`.



split	block_number	log_index	block_time	rounded_activated_at	rn	active
7,585	2,871,838	18	2024-11-09 19:18:00.000	2024-11-17 00:00:00.000	1	[v]
7,355	2,916,530	16	2024-11-19 17:08:12.000	2024-11-27 00:00:00.000	2	[ ]

Why This Happens

The current logic compares the `rounded_activated_at` of the second take rate change to the `rounded_activated_at` of the first take rate change:

```
-- rn is orderd by block and log_index, so this should encapsulate rops2 occurring afer rops
and rops.rn > rops2.rn
-- only find the next split that overlaps with the current one
and rops2.rounded_activated_at <= rops.rounded_activated_at
```

Since `rounded_activated_at` of the second take rate (`2024-11-27`) will **always be equal to or greater** than the first take rate's `rounded_activated_at` (`2024-11-17`), the condition is always satisfied.

As a result, the `CASE` statement marks the second take rate as **inactive (`false`)**.

2.6 Recommended Fix

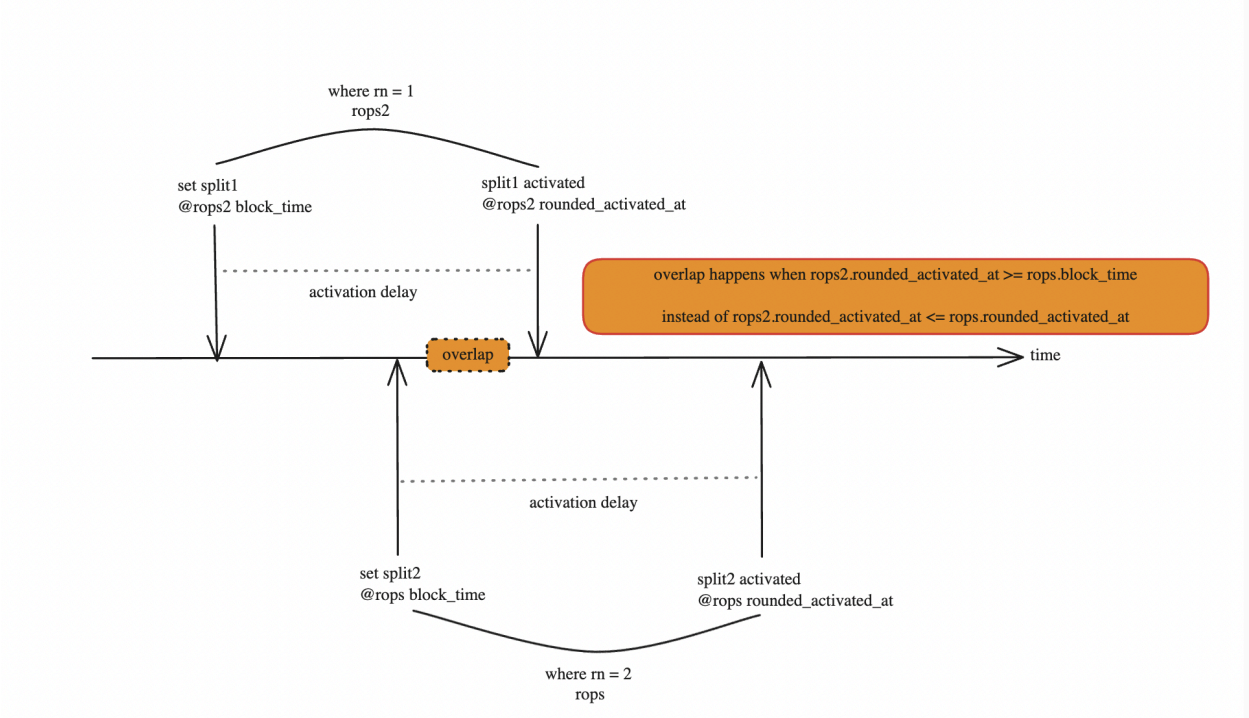
To fix this issue, update the comparison logic to compare:

- `rops2.rounded_activated_at` (previous activation time) with

- `rops.block_time` (current take rate's block time).

Corrected Join Condition:

```
, decorated_operator_avs_splits as (
    select
        rops.*,
        -- if there is a row, we have found another split that overlaps the current split
        -- meaning the current split should be discarded
        case
            when rops2.block_time is not null then false
            else true
         end as active
    from ranked_operator_avs_split_records as rops
    left join ranked_operator_avs_split_records as rops2 on (
        rops.operator = rops2.operator
        and rops.avs = rops2.avs
        -- rn is orderd by block and log_index, so this should encapsulate rops2 occurring afer rops
        and rops.rn > rops2.rn
        -- only find the next split that overlaps with the current one
        and rops2.rounded_activated_at >= rops.block_time
    )
)
```



3. Lack of Refund Mechanism for Unallocated Staker Incentives in Operator-Directed Rewards

3.1 Risk Level: Low

3.2 Status: Acknowledged *(The team decided not to make any code changes but to warn the AVSs about this edge case instead.)*

3.3 Root Cause

The current reward distribution mechanism has an edge case where an AVS incentivizes an operator for specific strategies, but the operator has no delegated shares in any of the specified strategies. Under this scenario:

- As long as there is at least one staker who has delegated a single share to the operator for any of the specified strategies, rewards will be distributed correctly due to the stake weight calculation.
- However, if an operator has no delegated shares across all specified strategies in the incentive campaign:
- The operator would still receive their allocated share of the rewards (e.g., 10% of the rewards deposited).
- The remaining portion of the rewards (e.g., 90%) would remain unallocated, as no stakers are eligible to claim these rewards.

Currently, there is no mechanism to refund the unallocated tokens back to the AVS. Refunds only occur if the operator was not registered with the AVS for the full duration of the incentive campaign. This lack of a refund mechanism leaves the unallocated tokens locked and inaccessible.

3.4 Impact

1. Unutilized Tokens:

- Tokens intended for stakers will remain unallocated even though operators receive their share.

4. Operator-Directed Reward Calculation Relies on Criteria Enforced by Smart Contracts

1.1 Risk Level: Low

1.2 Status: Acknowledged, no immediate risk, but noted for future consideration.

The Rewards V2 update introduces a mechanism where AVSs can directly incentivize operators based on their off-chain calculations. While this provides flexibility, it also introduces a reliance on the enforcement of criteria by the governing smart contract. The current implementation ensures that AVSs cannot publish a new reward submission with a campaign end timestamp that is later than the on-chain submission timestamp.

However, this design introduces a potential risk: if the smart contract logic enforcing these rules is improperly modified or bypassed in the future, it could lead to rewards being calculated incorrectly. Currently, the calculation of operator-directed rewards relies on the rewards submission always being retroactive. A smart contract update that removes the criteria ensuring retroactive rewards could cause operator-directed rewards to be calculated inaccurately.

We identify this as a potential future risk, which does not affect the current implementation.

5. Small Typo in Rewards Files

1.1 Risk Level: Low

1.2 Status: Fixed

There are minor typos in the file names under the `sidecar/pkg/rewards` directory, which could lead to potential confusion or inconsistencies in file references during development or debugging.

1. **`operatorAvsRegistrationSnaphots.go`**
    - Corrected Name: `operatorAvsRegistrationSnapshots.go`
2. **`operatorAvsRegistrationSnaphots_test.go`**
    - Corrected Name: `operatorAvsRegistrationSnapshots_test.go`

4. Retest

In the retest phase of the EigenLayer Rewards V2 Audit, the fixes for the audit findings and the implementation of the newly introduced default operator splits feature were reviewed.

4.1. Scope of the Retest

The retest was conducted to verify the resolution of issues identified during the initial audit of EigenLayer Rewards V2 and to assess the implementation of newly introduced functionality for default operator splits. Specifically, the scope included:

1. Retesting fixes made for the following audit findings:
    - End_time Calculation Error in Snapshots (SQL logic change)
    - Incorrect Handling of Take Rate Changes (SQL logic change)
    - Minor Typos in Rewards Files (Filename corrections)
2. Evaluating the implementation of the new feature for default operator splits, including:
    - Generation of `default_operator_split_snapshots`.
    - Integration of these snapshots into the relevant rewards tables:
        - AVS Rewards
        - Programmatic Incentives

        - Operator-Directed Rewards (Operators)
        - Operator-Directed Rewards (Stakers)

4.2. Retest Results

1. Retest of Audit Findings:
    - End_time Calculation Error in Snapshots: The issue involving the exclusion of the last day from time windows due to a subtraction error was resolved. We confirmed that the updated SQL logic now correctly includes all intended days in the snapshot generation. No further issues were found.
    - Incorrect Handling of Take Rate Changes: The logic determining the active status of take rate changes was corrected to use appropriate conditions. We verified that subsequent take rate changes are now properly flagged as active or inactive according to the intended 7-day delay mechanism. The issue has been fully resolved.
    - Minor Typos in Rewards Files: The typos in filenames within the `sidecar/pkg/rewards` directory have been corrected. The updated filenames now align with standard naming conventions, and no further issues were identified.
2. Evaluation of Default Operator Splits Implementation:
    - The newly implemented functionality for default operator splits was reviewed and tested. This included the generation of `default_operator_split_snapshots` and their integration into the following rewards tables:
        - AVS Rewards
        - Programmatic Incentives
        - Operator-Directed Rewards (Operators)
        - Operator-Directed Rewards (Stakers)

- The logic for incorporating default operator splits was verified to function as expected. The resulting snapshots and rewards calculations were consistent with the intended design. No issues were found during this evaluation.

5. Conclusion

The audit of the Rewards V2 calculation logic, Sidecar V1 and V2 integrations, the retest of the audit findings, and the evaluation of the new default operator splits functionality has been completed. All identified issues from the initial audit have been successfully resolved, and the newly implemented default operator splits functionality is working as intended. No further issues were identified.

OpenBlock Labs is satisfied with the quality of the fixes and the new implementation for default operator splits. We recommend proceeding with these changes as part of a new release for Sidecar.