Blobstream SP1

Security Assessment

August 20th, 2024 — Prepared by OtterSec

James Wang james.wang@osec.io

Robert Chen r@osec.io

Table of Contents

Executive Summary **2**

Overview 2

Key Findings 2

Scope **3**

Findings **4**

Vulnerabilities **5**

OS-BSP-ADV-00 | Lack of Block Height Validation 6

General Findings **7**

OS-BSP-SUG-00 | Removal of Deprecated Code 8

OS-BSP-SUG-01 | Documenting Validator Count Invariant 9

Appendices

Vulnerability Rating Scale **10**

Procedure **11**

© 2024 Otter Audits LLC. All Rights Reserved.

01 — Executive Summary

Overview

Celestia Labs and Succinct Labs engaged OtterSec to assess the `sp1-blobstream` program. This assessment was conducted between August 12th and August 14th, 2024. For more information on our auditing methodology, refer to Appendix B.

Key Findings

We produced 3 findings throughout this audit engagement.

In particular, we identified a critical vulnerability concerning the failure to validate `trusted_block_height` and `target_block_height`. This issue may result in committing incorrect states in `commitHeaderRange`, thereby bricking the Blobstream contract (OS-BSP-ADV-00).

We also made recommendations for removing multiple stale errors and events for better maintainability and clarity (OS-BSP-SUG-00), and suggested documenting Celestia's assumption of having 256 or fewer validators to prevent future changes in Celestia's validator count from breaking the contract (OS-BSP-SUG-01).

© 2024 Otter Audits LLC. All Rights Reserved.

02 — Scope

The source code was delivered to us in a Git repository at https://github.com/succinctlabs/sp1-blobstream. This audit was performed against commit ed6d066 and consisted of the following modules:

1. `program/src/main.rs`
2. `primitives/src/lib.rs`
3. `primitives/src/types.rs`
4. `contracts/src/SP1Blobstream.sol`

A brief description of the programs is as follows:

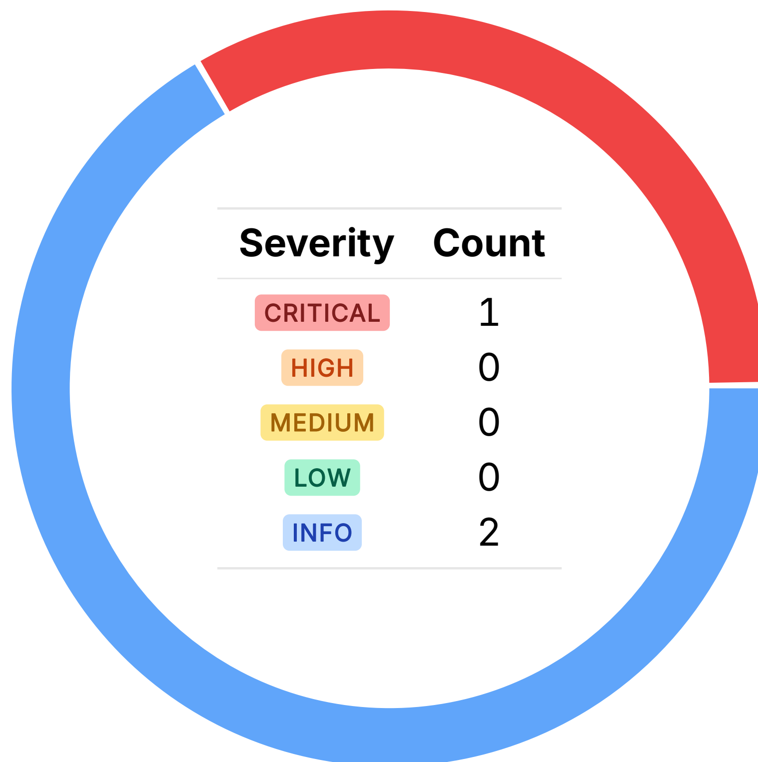
Name	Description
sp1-blobstream	The SP1Blobstream program and contracts allow Celestia to submit zkproofs of its block `data_root` onto Ethereum. The data roots may then be utilized by other protocols relying on Celestia as the data availability layer to perform on-chain validation of data inclusion.

© 2024 Otter Audits LLC. All Rights Reserved.

03 — Findings

Overall, we reported 3 findings.

We split the findings into **vulnerabilities** and **general findings**. Vulnerabilities have an immediate impact and should be remediated as soon as possible. General findings do not have an immediate impact but will aid in mitigating future vulnerabilities.



Severity	Count
CRITICAL	1
HIGH	0
MEDIUM	0
LOW	0
INFO	2

© 2024 Otter Audits LLC. All Rights Reserved.

04 — Vulnerabilities

Here, we present a technical analysis of the vulnerabilities we identified during our audit. These vulnerabilities have *immediate* security implications, and we recommend remediation as soon as possible.

Rating criteria can be found in Appendix A.

ID	Severity	Status	Description
OS-BSP-ADV-00	CRITICAL	RESOLVED	The failure to validate `trusted_block_height` and `target_block_height` may result in committing incorrect states in `commitHeaderRange`, potentially bricking the Blobstream contract.

© 2024 Otter Audits LLC. All Rights Reserved.

Lack of Block Height Validation CRITICAL OS-BSP-ADV-00

Description

In `SP1Blobstream`, `commitHeaderRange` commits new block headers and associated data commitments to the contract state. The block heights, `trusted_block_height` and `target_block_height`, are crucial as they indicate the specific points in the blockchain that are being validated and committed.

>_ *sp1-blobstream/program/src/main.rs* RUST

```rust
fn main() {
    [...]
    let ProofInputs {
        trusted_block_height,
        target_block_height,
        trusted_light_block,
        target_light_block,
        headers,
    } = proof_inputs;
    [...]
    let proof_outputs = ProofOutputs::abi_encode(&(
        [...]
        trusted_block_height,
        target_block_height,
        [...]
    ));
    sp1_zkvm::io::commit_slice(&proof_outputs);
}
```

In the current implementation, these block heights are passed as separate parameters within the proof input to the main function of the `SP1` program. However, these heights are not validated against the actual heights of the headers contained within the `trusted_light_block` and `target_light_block`. This lack of validation creates a vulnerability where an attacker may manipulate these block height values within the proof, resulting in committing an incorrect state to the contract.

Remediation

Utilize `trusted_light_block.signed_header.header.height.value` to fetch the `trusted_block_height` and `target_light_block.signed_header.header.height.value` to fetch the `target_block_height` instead of passing heights as separate inputs.

Patch

Resolved in e529c13.

© 2024 Otter Audits LLC. All Rights Reserved.

05 — General Findings

Here, we present a discussion of general findings during our audit. While these findings do not present an immediate security impact, they represent anti-patterns and may result in security issues in the future.

ID	Description
OS-BSP-SUG-00	Recommendation to remove deprecated events and errors.
OS-BSP-SUG-01	Suggestion to document Celestia’s assumption of having 256 or fewer validators to prevent future changes in Celestia’s validator count from breaking the contract.

© 2024 Otter Audits LLC. All Rights Reserved.

Removal of Deprecated Code

OS-BSP-SUG-00

Description

In `ISP1Blobstream`, the following events and errors are deprecated and may be removed to improve readability and maintainability:

1. `error LatestHeaderNotFound()`
2. `error DataCommitmentNotFound()`
3. `event HeaderRangeRequested([...])`
4. `event NextHeaderRequested([...])`

Remediation

Remove the events and errors mentioned above from the contract.

Patch

Resolved in e529c13.

© 2024 Otter Audits LLC. All Rights Reserved.

Documenting Validator Count Invariant

OS-BSP-SUG-01

Description

In `sp1-blobstream::get_validator_bitmap_commitment`, there is an implicit assumption about the maximum number of validators that may exist in the Celestia blockchain. The function is designed to handle up to 256 validators, as evidenced by the fixed-size bitmap ([bool; 256]) utilized to represent the validator set.

>_ *sp1-blobstream/program/src/main.rs* RUST

```rust
/// Construct a bitmap of the intersection of the validators that signed off on the trusted and
/// target header. Use the order of the validators from the trusted header. Equivocates slashing
    ↪  in
/// the case that validators are malicious. 256 is chosen as the maximum number of validators as
    ↪  it
/// is unlikely that Celestia has >256 validators.
fn get_validator_bitmap_commitment(
    trusted_light_block: &LightBlock,
    target_light_block: &LightBlock,
) -> U256 {
```

This assumption is based on the current configuration of the Celestia blockchain, where the number of validators is around 100, well within the 256-validator limit. However, it would be beneficial to explicitly document these additional requirements to prevent future changes to Celestia chain configurations from breaking the contract.

Remediation

Clearly document that the `sp1-blobstream` assumes a maximum of 256 validators.

Patch

Resolved in e529c13.

© 2024 Otter Audits LLC. All Rights Reserved.

A — Vulnerability Rating Scale

We rated our findings according to the following scale. Vulnerabilities have immediate security implications. Informational findings may be found in the General Findings.

CRITICAL

Vulnerabilities that immediately result in a loss of user funds with minimal preconditions.

Examples:

- Misconfigured authority or access control validation.
- Improperly designed economic incentives leading to loss of funds.

HIGH

Vulnerabilities that may result in a loss of user funds but are potentially difficult to exploit.

Examples:

- Loss of funds requiring specific victim interactions.
- Exploitation involving high capital requirement with respect to payout.

MEDIUM

Vulnerabilities that may result in denial of service scenarios or degraded usability.

Examples:

- Computational limit exhaustion through malicious input.
- Forced exceptions in the normal user flow.

LOW

Low probability vulnerabilities, which are still exploitable but require extenuating circumstances or undue risk.

Examples:

- Oracle manipulation with large capital requirements and multiple transactions.

INFO

Best practices to mitigate future security risks. These are classified as general findings.

Examples:

- Explicit assertion of critical internal invariants.
- Improved input validation.

© 2024 Otter Audits LLC. All Rights Reserved.

B — Procedure

As part of our standard auditing procedure, we split our analysis into two main sections: design and implementation.

When auditing the design of a program, we aim to ensure that the overall economic architecture is sound in the context of an on-chain program. In other words, there is no way to steal funds or deny service, ignoring any chain-specific quirks. This usually requires a deep understanding of the program's internal interactions, potential game theory implications, and general on-chain execution primitives.

One example of a design vulnerability would be an on-chain oracle that could be manipulated by flash loans or large deposits. Such a design would generally be unsound regardless of which chain the oracle is deployed on.

On the other hand, auditing the program's implementation requires a deep understanding of the chain's execution model. While this varies from chain to chain, some common implementation vulnerabilities include reentrancy, account ownership issues, arithmetic overflows, and rounding bugs.

As a general rule of thumb, implementation vulnerabilities tend to be more "checklist" style. In contrast, design vulnerabilities require a strong understanding of the underlying system and the various interactions: both with the user and cross-program.

As we approach any new target, we strive to comprehensively understand the program first. In our audits, we always approach targets with a team of auditors. This allows us to share thoughts and collaborate, picking up on details that others may have missed.

While sometimes the line between design and implementation can be blurry, we hope this gives some insight into our auditing procedure and thought process.

© 2024 Otter Audits LLC. All Rights Reserved.