Zellic

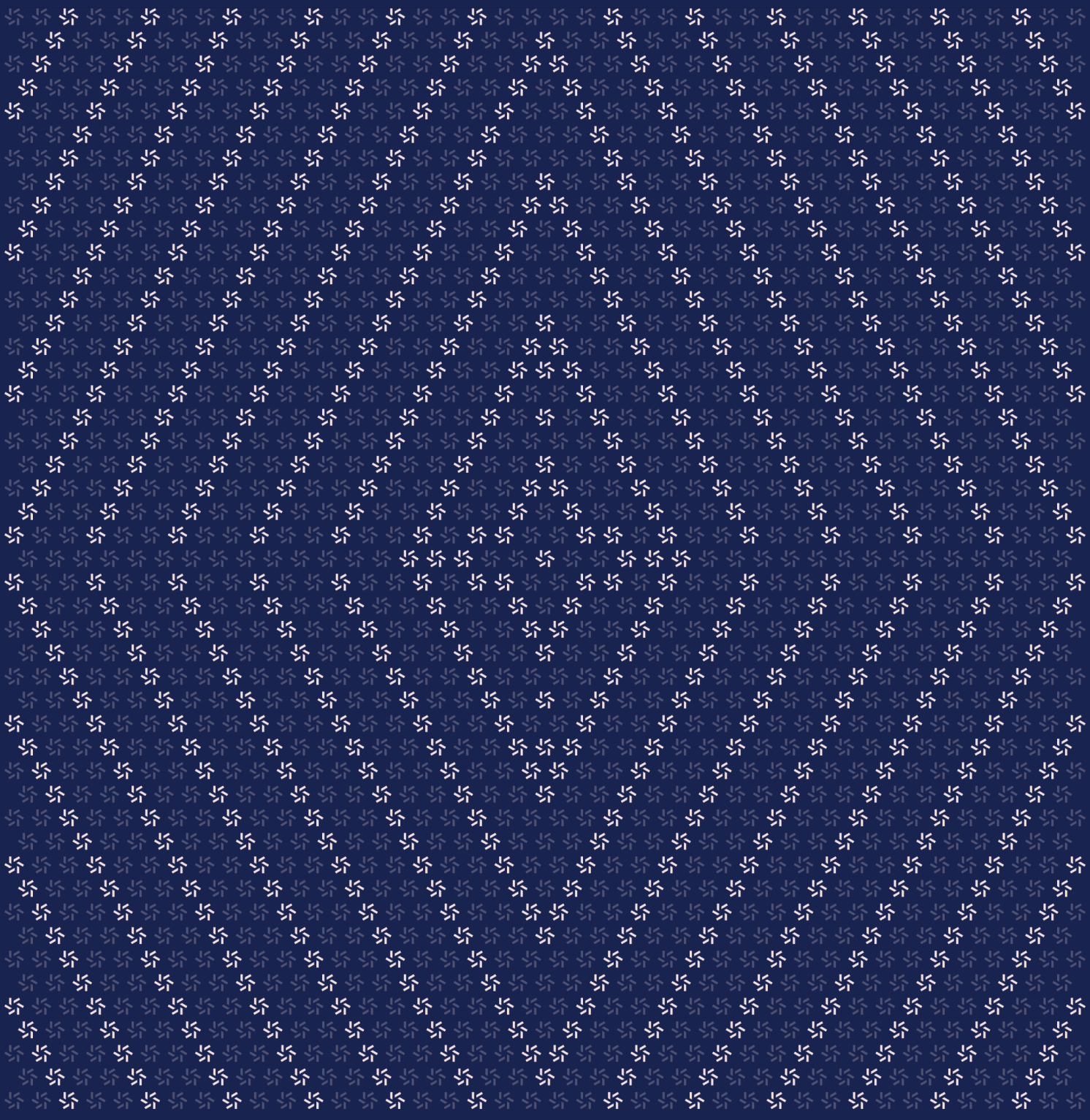
Prepared for
Meir Bank
David Mass
Hydrogen Labs

Prepared by
Juchang Lee
Jinheon Lee
Zellic

June 10, 2025

Rover Protocol

Smart Contract Patch Review



Contents

About Zellic

Zellic is a vulnerability research firm with deep expertise in blockchain security. We specialize in EVM, Move (Aptos and Sui), and Solana as well as Cairo, NEAR, and Cosmos. We review L1s and L2s, cross-chain protocols, wallets and applied cryptography, zero-knowledge circuits, web applications, and more.

Prior to Zellic, we founded the #1 CTF (competitive hacking) team ↗ worldwide in 2020, 2021, and 2023. Our engineers bring a rich set of skills and backgrounds, including cryptography, web security, mobile security, low-level exploitation, and finance. Our background in traditional information security and competitive hacking has enabled us to consistently discover hidden vulnerabilities and develop novel security research, earning us the reputation as the go-to security firm for teams whose rate of innovation outpaces the existing security landscape.

For more on Zellic's ongoing security research initiatives, check out our website zellic.io ↗ and follow @zellic_io ↗ on Twitter. If you are interested in partnering with Zellic, contact us at hello@zellic.io ↗.

1. Overview

1.1. Executive Summary

Zellic conducted a security assessment for Hydrogen Labs from June 4th to June 5th, 2025. During this engagement, Zellic reviewed Rover Protocol's code for security vulnerabilities, design issues, and general weaknesses in security posture.

1.2. Goals of the Assessment

In a security assessment, goals are framed in terms of questions that we wish to answer. These questions are agreed upon through close communication between Zellic and the client. In this assessment, we sought to answer the following questions:

- What are the features introduced between the previous commit and the current commit?
- What are the security implications of the new features?
- Does the new code introduce any additional security risks?

1.3. Non-goals and Limitations

We did not assess the following areas that were outside the scope of this engagement:

- Front-end components
- Infrastructure relating to the project
- Key custody
- Code that is already deployed

Due to the time-boxed nature of security assessments in general, there are limitations in the coverage an assessment can provide.

1.4. Results

During our assessment on the scoped Rover Protocol contracts, we discovered two findings. No critical issues were found. One finding was of high impact and one was of medium impact.

Breakdown of Finding Impacts

Impact Level	Count
Critical	0
High	1
Medium	1
Low	0
Informational	0

2. Introduction

2.1. About Rover Protocol

Hydrogen Labs contributed the following description of Rover Protocol:

> Rover is a liquid staking protocol for Botanix. Rover allows users on Botanix to participate in staking and earn yield on their Bitcoin, while remaining liquid via a Liquid Staking Token (LST).

2.2. Methodology

During a security assessment, Zellic works through standard phases of security auditing, including both automated testing and manual review. These processes can vary significantly per engagement, but the majority of the time is spent on a thorough manual review of the entire scope.

Alongside a variety of tools and analyzers used on an as-needed basis, Zellic focuses primarily on the following classes of security and reliability issues:

Basic coding mistakes. Many critical vulnerabilities in the past have been caused by simple, surface-level mistakes that could have easily been caught ahead of time by code review. Depending on the engagement, we may also employ sophisticated analyzers such as model checkers, theorem provers, fuzzers, and so on as necessary. We also perform a cursory review of the code to familiarize ourselves with the contracts.

Business logic errors. Business logic is the heart of any smart contract application. We examine the specifications and designs for inconsistencies, flaws, and weaknesses that create opportunities for abuse. For example, these include problems like unrealistic tokenomics or dangerous arbitrage opportunities. To the best of our abilities, time permitting, we also review the contract logic to ensure that the code implements the expected functionality as specified in the platform's design documents.

Integration risks. Several well-known exploits have not been the result of any bug within the contract itself; rather, they are an unintended consequence of the contract's interaction with the broader DeFi ecosystem. Time permitting, we review external interactions and summarize the associated risks: for example, flash loan attacks, oracle price manipulation, MEV/sandwich attacks, and so on.

Code maturity. We look for potential improvements in the codebase in general. We look for violations of industry best practices and guidelines and code quality standards. We also provide suggestions for possible optimizations, such as gas optimization, upgradability weaknesses, centralization risks, and so on.

For each finding, Zellic assigns it an impact rating based on its severity and likelihood. There is no hard-and-fast formula for calculating a finding's impact. Instead, we assign it on a case-by-case basis based on our judgment and experience. Both the severity and likelihood of an issue affect its impact. For instance, a highly severe issue's impact may be attenuated by a low likelihood.

We assign the following impact ratings (ordered by importance): Critical, High, Medium, Low, and Informational.

Zellic organizes its reports such that the most important findings come first in the document, rather than being strictly ordered on impact alone. Thus, we may sometimes emphasize an "Informational" finding higher than a "Low" finding. The key distinction is that although certain findings may have the same impact rating, their *importance* may differ. This varies based on various soft factors, like our clients' threat models, their business needs, and so on. We aim to provide useful and actionable advice to our partners considering their long-term goals, rather than a simple list of security issues at present.

2.3. Scope

The engagement involved a review of the following targets:

Rover Protocol Contracts

Type	Solidity
Platform	EVM-compatible
Target	rover-contracts
Repository	https://github.com/Hydrogen-Labs/rover-contracts ↗
Version	`Diff from e99a3171 to ea4d1488`
Programs	`packages/foundry/contracts/**.sol`

2.4. Project Overview

Zellic was contracted to perform a security assessment for a total of four person-days. The assessment was conducted by two consultants over the course of two calendar days.

Contact Information

The following project managers were associated with the engagement:

Jacob Goreski
Engagement Manager
jacob@zellic.io

Chad McDonald
Engagement Manager
chad@zellic.io

The following consultants were engaged to conduct the assessment:

Juchang Lee
Engineer
lee@zellic.io

Jinheon Lee
Engineer
jinheon@zellic.io

2.5. Project Timeline

The key dates of the engagement are detailed below.

June 4, 2025	Start of primary review period
June 5, 2025	End of primary review period

3. Detailed Findings

3.1. Reward distribution is unfair

Target	RovBTC		
Category	Business Logic	**Severity**	High
Likelihood	High	**Impact**	High

Description

The code below shows the current fee structure from collecting fees in shares.

```
function withdraw(uint256 assets, address receiver, address owner)
    public override nonReentrant notPaused returns (uint256) {
    // [...]
    uint256 feeAssets = _calculateRewardsFee(assetsWithdrawn, shares);

    if (feeAssets > 0) {
        IERC20(asset()).safeTransfer(rewardsFeeReceiver, feeAssets);
        emit RewardFeeCollected(assetsWithdrawn - shares, feeAssets);
    }

    uint256 assetsForUser = assetsWithdrawn - feeAssets;

    // [...]
}
```

In an ERC-4626 Vault, the value of a share is not fixed: over time and as you earn, the amount of principal assets that can be exchanged for 1 share gradually increases.

The code overlooks this and puts shares in place of principal when calling the `_calculateRewardsFee(assetsWithdrawn, shares)` function.

This approach works correctly when initially 1 share = 1 asset, but as time passes and profits accumulate (e.g., 1 share = 1.2 assets), the number of shares and the actual principal assets diverge. Consequently, the formula `assetsForUser = assetsWithdrawn - feeAssets` becomes inaccurate, failing to properly account for the increased asset value per share.

Impact

This may result in users being unfairly charged higher fees.

Recommendations

Instead of charging fees on shares, collect fees by tracking assets.

Remediation

This issue has been acknowledged by Hydrogen Labs, and fixes were implemented in the following commits:

- 637f31dc ↗
- dc072db7 ↗

Hydrogen Labs brought this issue to our attention prior to the submission of the official report. During our review of the proposed fix, we discovered a remaining issue, detailed in Finding 3.2. ↗.

3.2. Reward tracking not updated for transfer

Target	RovBTC		
Category	Coding Mistakes	**Severity**	Medium
Likelihood	Medium	**Impact**	Medium

Description

This issue was identified during the remediation process and has been included in the report for completeness.

The `principalAssets` mapping, introduced in commit `f935d158` as a remediation to finding 3.1., is used to track a user's assets. It's updated during `depositStakedBTC`, `_deposit`, `withdraw`, `redeem`, and `redeemNativeBTC`, handling both deposits and withdrawals appropriately.

However, transfers do not update `principalAssets`. This oversight can lead to inconsistencies and potential issues in tracking user balances.

Impact

Since `principalAssets` don't follow the tokens when they're transferred, a mismatch arises between the token's current holder and the principal depositor. This represents a fundamental flaw in the accounting system.

As a result, the recipient may end up paying disproportionately high fees on revenue they didn't earn, leading to a direct financial loss. Conversely, the sender of the token avoids paying fees on their own revenue.

Recommendations

We recommend overriding `_update` to ensure that `principalAssets` is updated on transfer as well.

Remediation

This issue has been acknowledged by Hydrogen Labs, and a fix was implemented in commit `dc072db7`.

4. System Design

This section outlines the high-level architecture of the system and how its core components interact. It captures key behaviors such as asset deposit and withdrawal mechanics, reward distribution, administrative controls, and access role enforcement. Special attention is given to externally accessible functions and configuration points that could influence system state or user balances.

Where applicable, the description highlights inputs that are user-controllable and examines how these could be exploited by a malicious actor to violate system integrity. It also notes critical invariants — such as value-locking limits, pause states, and role-based access control — that must be preserved to maintain secure operation.

Not all components or edge cases may be covered in this overview. The omission of a module or interaction does not imply its security or insignificance.

4.1. Rover BTC token

The RovBTC contract functions as an ERC-4626–compliant vault, using pBTC as its underlying asset and rovBTC as shares.

Key functionalities

These are its key functionalities.

- **Deposits.** It supports deposits of native BTC (autoconverts to pBTC), direct pBTC, and staked stBTC. All deposits are subject to total value locked (TVL) limits and contract pause status.
- **Withdrawals/redemptions.** It allows users to withdraw assets (pBTC or native BTC) or redeem rovBTC shares. These operations trigger staking-reward harvesting and apply a configurable rewards fee.
- **Rewards and fees.** A percentage of staking rewards (`rewardsFee`) is collected and sent to a designated `rewardsFeeReceiver`.
- **Admin mint/burn.** Authorized roles can mint or burn rovBTC tokens.
- **Configuration.** Administrators can set the maximum deposit TVL, rewards-fee percentage, and rewards-fee receiver and pause/unpause the contract.
- **Preview functions.** It provides standard ERC-4626 preview functions (`previewDeposit`, `previewWithdraw`, `previewRedeem`) to estimate outcomes.

4.2. Roles

The RoleManager contract centrally manages access permissions across the protocol using OpenZeppelin's AccessControlUpgradeable.

Defined roles

These are its defined roles.

- **RoleManager admin.** This manages roles within the RoleManager contract.
- **rovBTC minter/burner.** This permits minting and burning of rovBTC tokens.
- **rovBTC admin.** This controls key configurations of the RovBTC contract (e.g., TVL limits, reward fees, and reward receiver).
- **Deposit/withdraw pauser.** This authorizes pausing and unpausing of deposit/withdrawal operations on RovBTC.

4.3. Staking

The StakeManager contract serves as the main entry point for users to deposit and withdraw assets from the protocol. It tracks ownership via the rovBTC token.

Deposits

Users can deposit Bitcoin into the protocol, receiving rovBTC in return. This involves the following.

- **`depositBTC()`**. This function allows users to deposit Bitcoin without providing a referral ID.
- **`depositBTC(uint256 _referralId)`**. This function enables Bitcoin deposits with an optional `_referralId`. The amount of rovBTC minted is equal to the deposited `msg.value`.
- **TVL enforcement.** Before a deposit, the system calculates the `totalTVL`. If a `maxDepositTVL` is set (i.e., not 0) and the incoming deposit would cause the `totalTVL` to exceed `maxDepositTVL`, the transaction will revert with a `MaxTVLReached` error. The `calculateTVL()` function currently returns the contract's balance as the total value locked.
- **Pause functionality.** Deposits are restricted when the contract is in a paused state. If paused is true, the `depositBTC` function will revert with a `ContractPaused` error.
- **Events.** Upon a successful deposit, a `Deposit` event is emitted, including the depositor, amount, `rovBTCMinted`, and `referralId`.

Configuration

The StakeManager contract includes functions for administrative configuration:

- **`setPaused(bool _paused)`**. This function allows setting the contract's paused state, which controls deposit and withdrawal operations. It can only be called by an address with the `DEPOSIT_WITHDRAW_PAUSER` role.
- **`setMaxDepositTVL(uint256 _maxDepositTVL)`**. This function adjusts the maximum TVL allowed for deposits. Setting `_maxDepositTVL` to 0 disables the limit. Only

addresses with the `STAKE_MANAGER_ADMIN` role can call this function.

5. Diff Scope

The purpose of this section is to document the exact diffs of the codebase that were considered in scope for this audit.

We focused on the changes made between the last audit (e99a3171) and the current version (ea4d1488) of the codebase.

5.1. Core architecture changes

The existing StakeManager and RovBtcToken were merged to become the RovBTC contract. The responsibilities of depositing assets (previously StakeManager) and representing user shares (previously RovBtcToken) are now combined within RovBTC.

RovBTC is ERC-4626–based vault and directly mints/burns its own shares upon deposit/withdrawal.

This contract performs the following functions:

- It manages deposits of native BTC, wrapped pBTC, and staked stBTC.
- It handles withdrawals to pBTC or native BTC.
- It is the ERC-20 token representing shares in the vault (replacing the old RovBtcToken).

Two contracts have been added to implement this.

- pBTC — This contract is an ERC-20 token wrapped in native BTC.
- stBTC — This contract is ERC-4626 based and includes reward-distribution logic, which is the core staking vault for depositing pBTC and earning interest.

5.2. New features

This section outlines some of the new features.

1. Interest earnings and protocol-fee machine

Revenue generated by the stBTC vault is distributed to RovBTC users. In the process, RovBTC contracts are given the ability to collect a fee on the profits, called `rewardsFee`, which is a new revenue model for the protocol.

2. Support for multiple deposit and withdrawal paths

The following functions have been added.

- `depositBTC` — deposits native BTC (gas tokens) directly
- `depositPBTC` — deposits pBTC tokens
- `depositStakedBTC` — allows users who already have stBTC to deposit it directly
- `redeem` — burns rovBTC and gets pBTC back

- `redeemNativeBTC` — burns rovBTC and gets native BTC back directly for user convenience

5.3. Role-management updates

Changes have been made to role management.

- The `isStakeManagerAdmin` has been renamed to `isRovBTCAdmin`.
- RoleManager's `initialize` function's visibility was changed from public to external.
- The `STAKE_MANAGER_ADMIN` role constant was renamed to `ROVBTC_ADMIN`.
- The `NotStakeManagerAdmin` error was renamed to `NotRovBTCAdmin`.

6. Assessment Results

During our assessment on the scoped Rover Protocol contracts, we discovered two findings. No critical issues were found. One finding was of high impact and one was of medium impact.

6.1. Disclaimer

This assessment does not provide any warranties about finding all possible issues within its scope; in other words, the evaluation results do not guarantee the absence of any subsequent issues. Zellic, of course, also cannot make guarantees about any code added to the project after the version reviewed during our assessment. Furthermore, because a single assessment can never be considered comprehensive, we always recommend multiple independent assessments paired with a bug bounty program.

For each finding, Zellic provides a recommended solution. All code samples in these recommendations are intended to convey how an issue may be resolved (i.e., the idea), but they may not be tested or functional code. These recommendations are not exhaustive, and we encourage our partners to consider them as a starting point for further discussion. We are happy to provide additional guidance and advice as needed.

Finally, the contents of this assessment report are for informational purposes only; do not construe any information in this report as legal, tax, investment, or financial advice. Nothing contained in this report constitutes a solicitation or endorsement of a project by Zellic.