Zellic

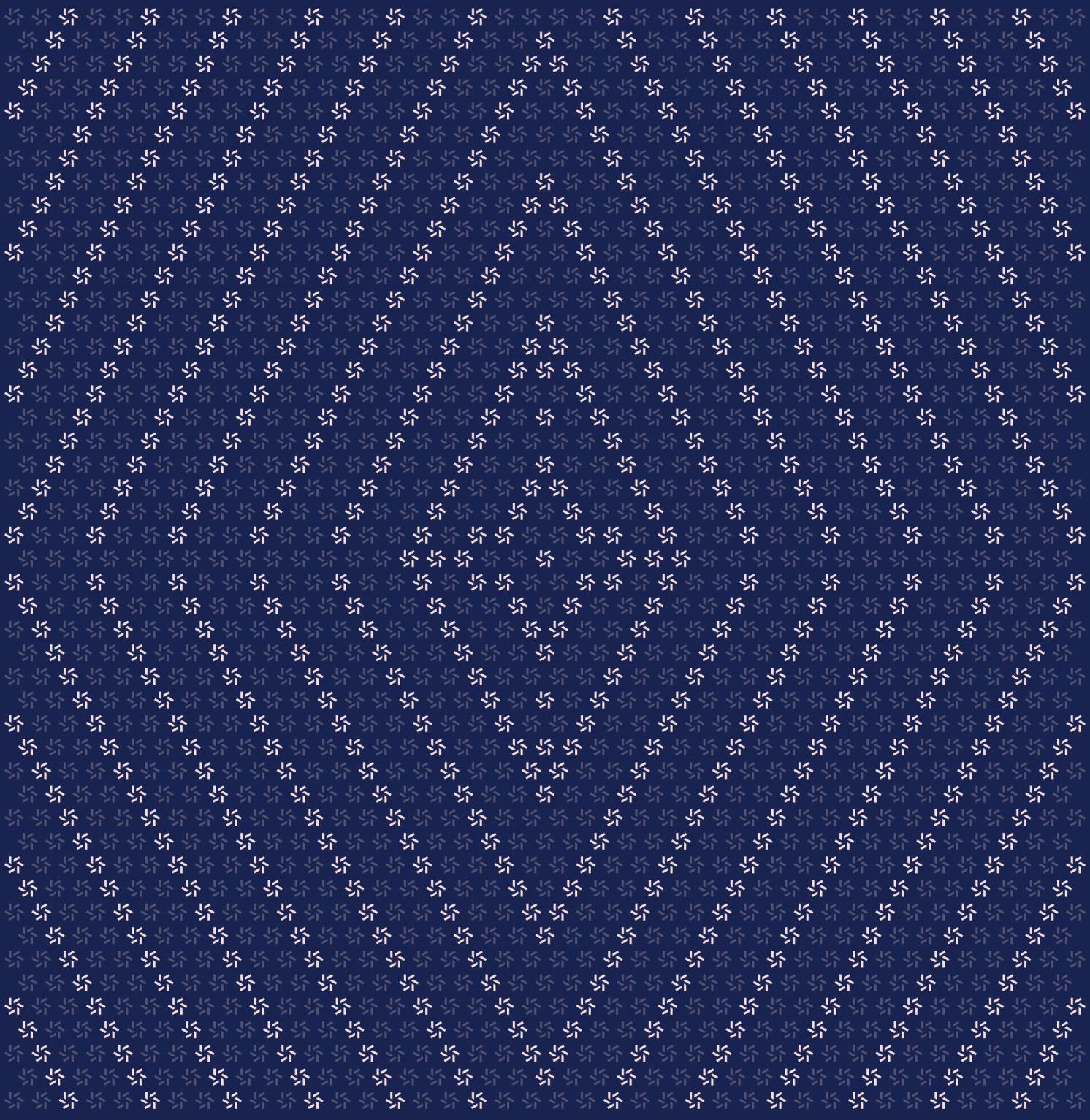
Prepared for
Meir Bank
David Mass
Hydrogen Labs

Prepared by
Jaeeu Kim
SeungJun Kim
Zellic

April 11, 2024

Rover

Smart Contract Security Assessment



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 on April 11th, 2024. During this engagement, Zellic reviewed Rover'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:

- Could a malicious user drain funds from the contract?
- Can an on-chain attacker cause a Denial-of-Service (DoS) attack on the contract?
- Is there a risk of unauthorized withdrawal of funds belonging to other users?

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

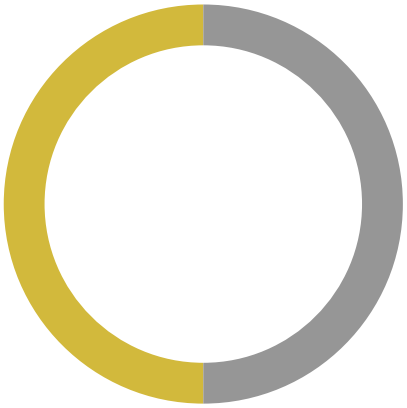
Due to the time-boxed nature of security assessments in general, there are limitations in the coverage an assessment can provide.

1.4. Results

During our assessment on the scoped Rover contracts, we discovered two findings. No critical issues were found. One finding was of medium impact and the other finding was informational in nature.

Breakdown of Finding Impacts

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



2. Introduction

2.1. About Rover

Hydrogen Labs contributed the following description of Rover:

> Hydrogen Labs Rover mints RoverBTC tokens equivalent to the deposit. The protocol allows users to stake native BTC on Botanix in exchange for rovBTC, a liquid staking token that represents a share of the protocol's total value locked.

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 Contracts

Repository	https://github.com/Hydrogen-Labs/rover-contracts ↗
Version	rover-contracts: `e99a31711d26bc221efaab6af3867e5d53f33bcf`
Programs	• AccessControl/IRoleManager.sol • AccessControl/RoleManager.sol • AccessControl/RoleManagerStorage.sol • Errors/Errors.sol • RoverBtcToken/IRovBtcToken.sol • RoverBtcToken/RovBtcToken.sol • RoverBtcToken/RovBtcTokenStorage.sol • StakeManager/IStakeManager.sol • StakeManager/StakeManager.sol • StakeManager/StakeManagerStorage.sol
Type	Solidity
Platform	EVM-compatible

2.4. Project Overview

Zellic was contracted to perform a security assessment with two consultants for a total of one person-day. The assessment was conducted over the course of one calendar day.

Contact Information

The following project manager was associated with the engagement:

Chad McDonald
Engagement Manager
chad@zellic.io

The following consultants were engaged to conduct the assessment:

Jaeeu Kim
Engineer
jaeeu@zellic.io

SeungJun Kim
Engineer
seungjun@zellic.io

2.5. Project Timeline

The key dates of the engagement are detailed below.

April 11, 2024	Start of primary review period
April 11, 2024	End of primary review period

3. Detailed Findings

3.1. No withdrawal functionality

Target	StakeManager		
Category	Coding Mistakes	**Severity**	High
Likelihood	Low	**Impact**	Medium

Description

In StakeManager, the withdrawal functionality is absent, preventing users from withdrawing their deposits.

Impact

Users' deposits may become locked within the contract.

Recommendations

Ensure that the ownership of the upgrading contract and the stages of the staking process are prominently documented so that users are aware of and accept the associated risks.

Remediation

This issue has been acknowledged by Hydrogen Labs.

According to Hydrogen Labs's response, they intentionally have no withdrawal function in this codebase, a similar approach to Lido's staking contracts pre-merge, as there is a similar dynamic with Botanix's staking development rollout — staking rollout will necessitate a V2 migration with withdrawals enabled.

3.2. Allowing users to deposit zero amount

Target	StakeManager		
Category	Coding Mistakes	**Severity**	Informational
Likelihood	Low	**Impact**	Informational

Description

In StakeManager, the `depositBTC` function is used to deposit users' funds. However, this function does not check if `msg.value` is zero. So, a user could call this function with zero amount of `msg.value`.

Impact

A malicious user could call the deposit function with zero amount, which would trigger a `Deposit` event emission. If there is an event tracker monitoring event emissions, it could lead to inaccuracies in event tracking.

Recommendations

Consider adding a require statement to check the amount of `msg.value`.

Remediation

This issue has been acknowledged by Hydrogen Labs, and a fix was implemented in commit `06a64a53` ↗.

4. Threat Model

This provides a full threat model description for various functions. As time permitted, we analyzed each function in the contracts and created a written threat model for some critical functions. A threat model documents a given function's externally controllable inputs and how an attacker could leverage each input to cause harm.

Not all functions in the audit scope may have been modeled. The absence of a threat model in this section does not necessarily suggest that a function is safe.

4.1. Owner functions

The owner is a trusted entity and has extensive control over the protocol. The owner should document this to ensure that the user is aware of it and accepts it.

The owner has the following abilities:

Granting roles

- Grant the `ROV_BTC_MINTER_BURNER` role.
  - Mint the Rover token.
  - Burn the Rover token.
- Grant the `STAKE_MANAGER_ADMIN` role.
  - Set the `maxDepositTVL`.
- Grant the `TOKEN_ADMIN` role.
  - Pause the Rover token.
- Grant the `DEPOSIT_WITHDRAW_PAUSER` role.
  - Pause the stake-manager contract.

Upgrading

- Upgrade contracts.

4.2. User functions

Users are unauthenticated. So users should only be able to use allowed functions.

The user has the following abilities:

Depositing

- Deposit BTC to protocol. If the amount of deposit is larger than `maxDepositTVL`, it fails to deposit.
- Mint Rover token as the amount of `msg.value`.

- Set the referral ID for deposit.

5. Assessment Results

At the time of our assessment, the reviewed code was not deployed to the Ethereum Mainnet.

During our assessment on the scoped Rover contracts, we discovered two findings. No critical issues were found. One finding was of medium impact and the other finding was informational in nature.

5.1. 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.