Zellic

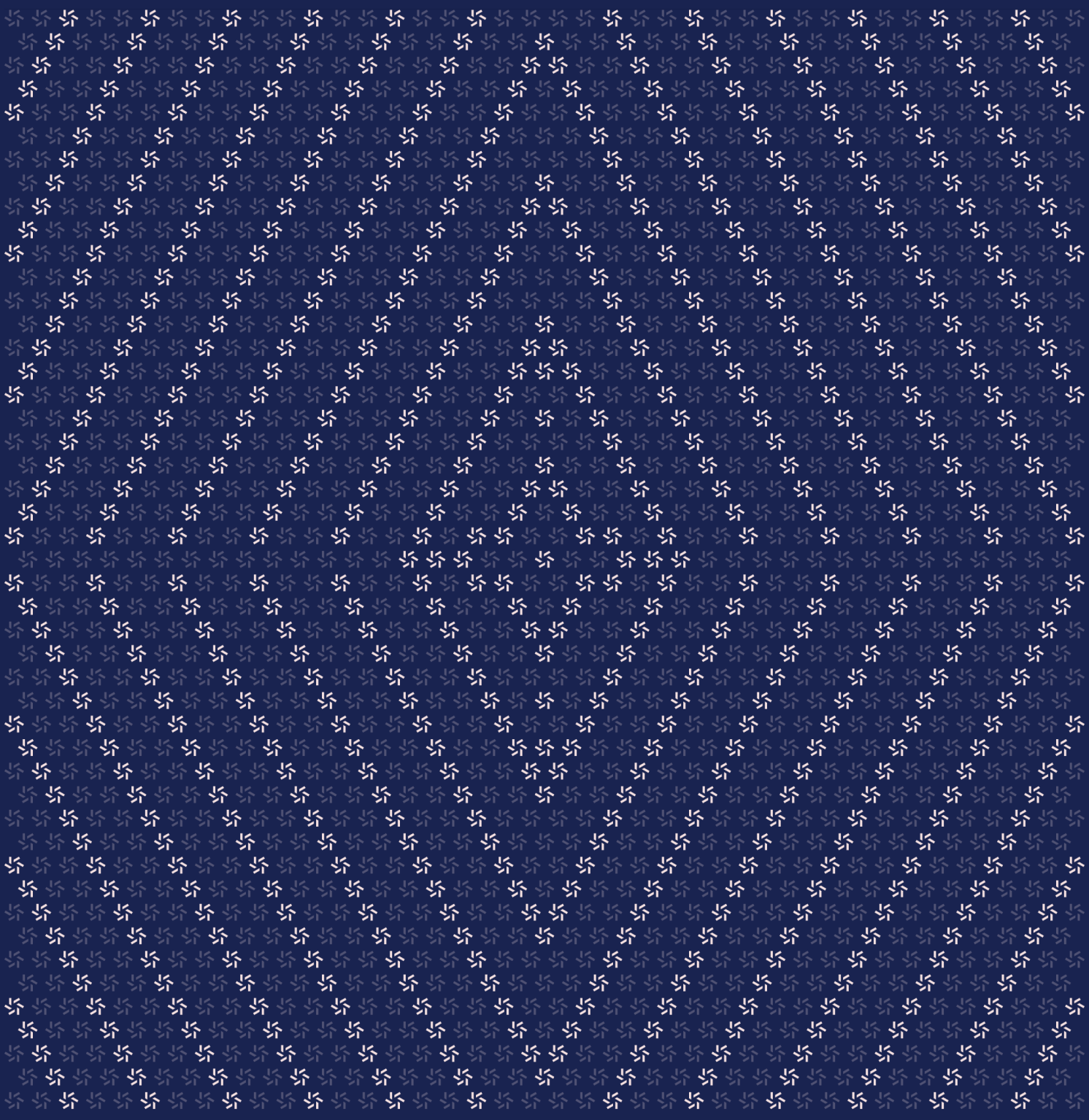
Prepared for
red
Infrared

Prepared by
Junghoon Cho
Vlad Toie
Zellic

October 10, 2024

Infrared Berachain Core Integration

Testing Suite Engagement



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 testing suite improvement for Infrared from September 25th to October 9th, 2024. During this engagement, Zellic adapted Infrared Berachain Core Integration's code and testing suite to improve both the quality and the coverage of the tests. Additionally, Zellic provided a comprehensive report detailing the changes made to the testing suite and the rationale behind them.

1.2. Goals of the Engagement

In this engagement, the following goals have been agreed upon through discussions with Infrared:

- Write test cases for `Infrared.sol:harvestBase()`.
- Write test cases for `Infrared.sol:harvestVault()`.
- Write test cases for `Infrared.sol:harvestBribes()`.
- Write test cases for `Infrared.sol:collectBribes()`.
- Write test cases for `Infrared.sol:harvestBoostRewards()`.
- Deploy the Berachain core contracts as base for the proof of liquidity harvest function tests.

1.3. Non-goals and Limitations

We did not assess nor engage in the following areas that were outside the scope of this engagement:

- Front-end components
- Infrastructure relating to the project
- Key custody
- The security of any of the contracts we have improved the tests for

We also did not define and implement invariants or fuzzing tests.

Due to the time-boxed nature of engagements in general, there are limitations in the coverage an engagement can provide.

2. Introduction

2.1. About Infrared Berachain Core Integration

Infrared contributed the following description of Infrared Berachain Core Integration:

> Infrared ↗ is focused on building infrastructure around the Proof of Liquidity (PoL) mechanism pioneered by Berachain. The protocol aims to maximize value capture by providing easy-to-use liquid staking solutions for BGT and BERA, node infrastructure, and PoL vaults. Through building solutions around PoL, Infrared is dedicated to enhancing the user experience and driving the growth of the Berachain ecosystem.

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.

Finally, Zellic provides a list of miscellaneous observations that do not have security impact or are not directly related to the scoped contracts itself. These observations — found in the Discussion (3. ↗) section of the document — may include suggestions for improving the codebase, or general recommendations, but do not necessarily convey that we suggest a code change.

2.3. Scope

The engagement involved an improvement of the testing suite for the following targets:

Infrared Berachain Core Integration Contracts

Type	Solidity
Platform	EVM-compatible
Target	infrared-contracts
Repository	https://github.com/infrared-dao/infrared-contracts/ ↗
Version	`bfc3d006d9275a136ba518e9375f51ca620f6bba`
Programs	`contracts/tests/unit/core/Infrared/*`

2.4. Project Overview

Zellic was contracted to perform testing suite improvement for a total of three person-weeks. The engagement was conducted by two consultants over the course of two calendar weeks.

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:

Junghoon Cho
Engineer
junghoon@zellic.io ↗

Vlad Toie
Engineer
vlad@zellic.io ↗

2.5. Project Timeline

The key dates of the engagement are detailed below.

September 25, 2024	Kick-off call
September 25, 2024	Start of primary review period
October 9, 2024	End of primary review period

3. Project Details

This section documents the current workflow of the project, as well as our notes and deliverables.

3.1. Workflow of Infrared Berachain Core Integration

During the planned work period, Zellic conducted improvements and rearrangements of the Infrared protocol's test suite, rather than a usual security assessment.

First, the main functionality of Infrared is as follows: when a user stakes assets in Infrared, Infrared stakes those assets on behalf of the user in Berachain's Rewards Vault through InfraredVault. Later, when InfraredVault harvests the BGT rewards received from the Rewards Vault, the user receives IBGT issued correspondingly as a reward.

The work requested by Infrared regarding the test suite for this implementation could be divided into two main tasks. First, the test suite version at that time was using mocks for all Berachain proof-of-liquidity contracts that Infrared interacts with, such as Rewards Vault and BGT. We were requested to replace these mocks with the actual Berachain contract implementations to perform more actual and accurate tests. The second task was to map the invariants of functions that harvest assets in Berachain PoL and to write new test scripts to achieve high test coverage. The list of these functions is as follows:

- `Infrared::harvestBase`
- `Infrared::harvestVault`
- `Infrared::harvestBribes`
- `Infrared::collectBribes`
- `Infrared::harvestBoostRewards`

Before working on the test suite, Zellic completed a comprehensive understanding of the codebase and defined several high-level invariants. Although this engagement was not a security assessment, we fully grasped the developers' intentions and communicated any security concerns that arose during the process to Infrared, aiming to eliminate any remaining minor security issues.

Subsequently, Zellic began the task of adapting Berachain's PoL core contracts to Infrared by replacing the existing mocks. During this process, there were compatibility issues such as overlapping in remapping and ABI mismatches, which were resolved after extensive work. Eventually, Zellic modified the existing Helper base contract used in the test contracts to inherit from POLTest, the base contract of Berachain's codebase test contracts, enabling the testing of Infrared's functionality alongside the setup of Berachain's PoL contracts.

We also rewrote the test contracts for functions that harvest assets received as rewards to align with the adaptation of Berachain contracts. We identified the invariants of these functions and wrote scripts to trigger each error, thereby increasing overall coverage.

In the end, Zellic successfully completed the improvement of Infrared's test suite in accordance with the client's request.

3.2. Removal of `bgt.setCommission` function call.

We have removed the call to the BGT `setCommission` function within the `_updateValidatorCommission`:

```
/// @notice Updates validator commission rate calling BGT to set.
function _updateValidatorCommission(bytes memory _pubkey, uint256 _commission)
    private {
    if (_commission > COMMISSION_MAX) revert Errors.InvalidCommissionRate();
    emit ValidatorCommissionUpdated(msg.sender, _pubkey,
    _getValidatorCommission(_pubkey), _commission);
    // _bgt.setCommission(_pubkey, _commission);
}
```

This change is in favor of the newly defined BGT route for updating commissions, which make use of the `queueCommissionChange` and `activateCommissionChange` functions:

```
/// @inheritdoc IBGT
function queueCommissionChange(bytes calldata pubkey, uint256 rate)
    external onlyOperator(pubkey) {
    if (rate > TEN_PERCENT) InvalidCommission.selector.revertWith();

    QueuedCommission storage c = queuedCommissions[pubkey];
    (c.blockNumberLast, c.rate) = (uint32(block.number), uint224(rate));
    emit QueueCommissionChange(pubkey, commissions[pubkey], rate);
}


/// @inheritdoc IBGT
function activateCommissionChange(bytes calldata pubkey) external {
    QueuedCommission storage c = queuedCommissions[pubkey];
    (uint32 blockNumberLast, uint224 rate) = (c.blockNumberLast, c.rate);
    // check if the commission is queued, if not revert with error
    if (blockNumberLast == 0) CommissionNotQueued.selector.revertWith();
    _checkEnoughTimePassed(blockNumberLast);

    commissions[pubkey] = rate;
    delete queuedCommissions[pubkey];
    emit ActivateCommissionChange(pubkey, rate);
}
```

This new mechanism ensures that any changes to the commission rate are first queued and then activated after a certain period. This delay enhances security by preventing immediate and potentially malicious commission changes, allowing stakeholders to react if necessary.

However, integrating this new commission update process requires significant design considera-

tions. It affects how commission rates are stored, updated, and validated across the system. Due to these substantial architectural implications, we have decided not to modify the existing code directly. We believe it's best to leave the implementation details to the Infrared team, who can carefully integrate these changes while considering the overall system design as well as the potential impact on other modules.

We recommend that Infrared team reviews these proposed changes and integrates them in a manner that aligns with the project's architectural principles and goals. Additionally, we recommend testing the new commission update mechanism thoroughly to ensure it behaves as expected and does not introduce ulterior issues.

3.3. 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.