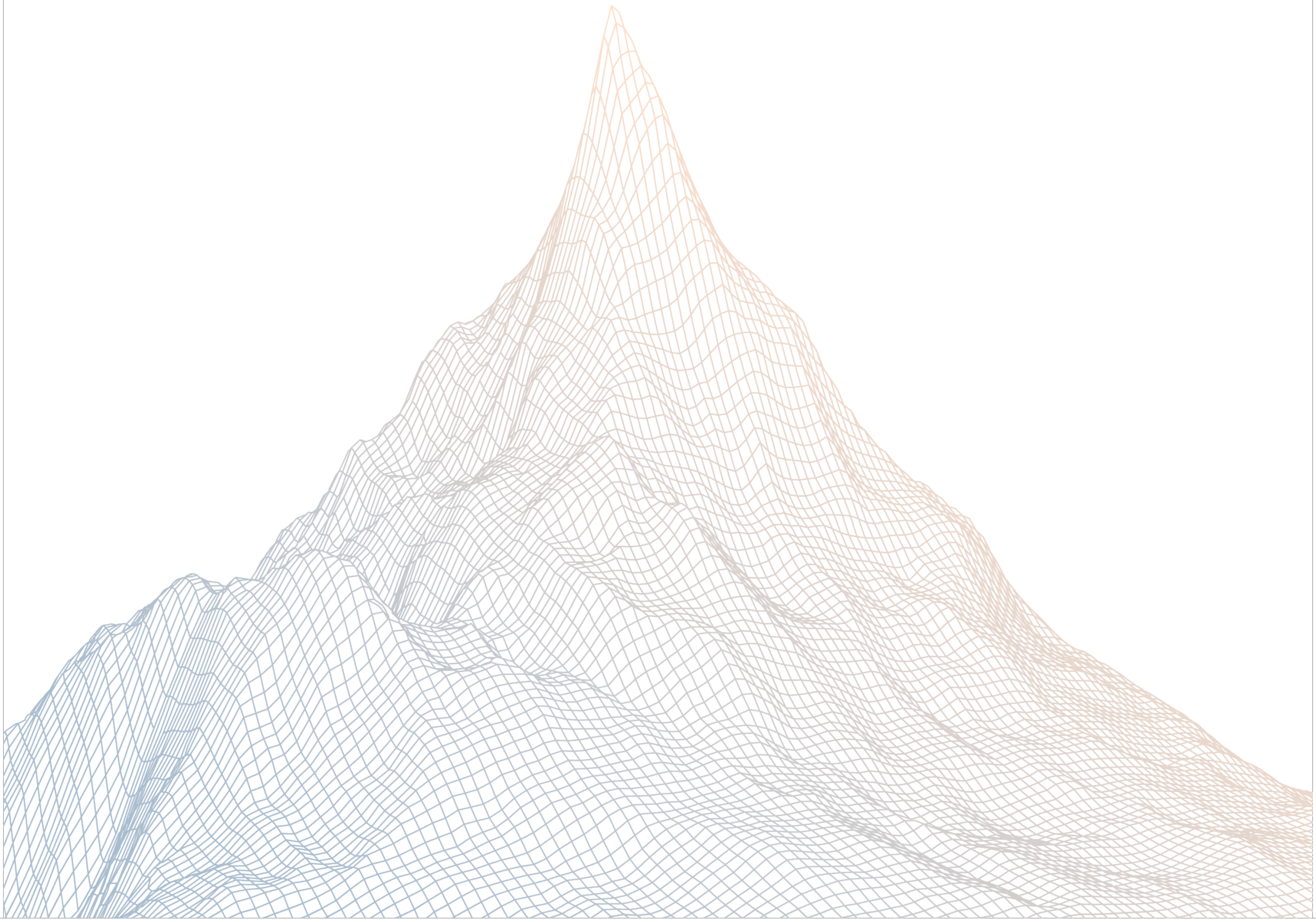
Zenith

Infrared

Smart Contract Security Assessment

VERSION 1.1



AUDIT DATES: April 4th to April 5th, 2025

AUDITED BY: Matte
ladboy

Contents

1 Introduction

1.1 About Zenith

Zenith is an offering by Code4rena that provides consultative audits from the very best security researchers in the space. We focus on crafting a tailored security team specifically for the needs of your codebase.

Learn more about us at https://code4rena.com/zenith.

1.2 Disclaimer

This report reflects an analysis conducted within a defined scope and time frame, based on provided materials and documentation. It does not encompass all possible vulnerabilities and should not be considered exhaustive.

The review and accompanying report are presented on an "as-is" and "as-available" basis, without any express or implied warranties.

Furthermore, this report neither endorses any specific project or team nor assures the complete security of the project.

1.3 Risk Classification

SEVERITY LEVEL	IMPACT: HIGH	IMPACT: MEDIUM	IMPACT: LOW
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

2

Executive Summary

2.1 About Infrared

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 vaults. Through building solutions around Proof of Liquidity (PoL), Infrared is dedicated to enhancing the user experience and driving the growth of the Berachain ecosystem.

2.2 Scope

The engagement involved a review of the following targets:

Target	infrared-contracts
Repository	https://github.com/infrared-dao/infrared-contracts
Commit Hash	`d13b64609c2668258a3d982e03e96e41cf9fc94c`
Files	`Changes in PR-596`

2.3 Audit Timeline

April 4, 2025	Audit start
April 5, 2025	Audit end
April 7, 2025	Report published

2.4 Issues Found

SEVERITY	COUNT
Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	0
Informational	2
Total Issues	**2**

3

Findings Summary

ID	Description	Status
I-1	Distributor address(0) check can be removed to save gas.	Resolved
I-2	Consider add make bgtIncentiveDistributor variable immutable and declare the variable storage in InfraredV1_2	Resolved

4 Findings

4.1 Informational

A total of 2 informational findings were identified.

[I-1] Distributor address(0) check can be removed to save gas.

SEVERITY: Informational	IMPACT: Informational
STATUS: Resolved	LIKELIHOOD: Low

Target

- InfraredV1_4

Description:

```
function claimBGTIncentives(
    IBGTIncentiveDistributor.Claim[] calldata _claims
) external onlyKeeper {

    address distributor = bgtIncentiveDistributor;
    if (distributor == address(0)) revert Errors.BGTDistributorNotSet();
    if (_claims.length == 0) revert Errors.InvalidArrayLength();

    IBGTIncentiveDistributor(distributor).claim(_claims);
```

The `distributor` is `bgtIncentiveDistributor` and `bgtIncentiveDistributor` cannot be address(0) because of the check in the initializer.

Recommendations:

```
    function claimBGTIncentives(
        IBGTIncentiveDistributor.Claim[] calldata _claims
    ) external onlyKeeper {

    address distributor = bgtIncentiveDistributor;
    if (distributor == address(0)) revert Errors.BGTDistributorNotSet();
```

```
        if (_claims.length == 0) revert Errors.InvalidArrayLength();

    IBGTIncentiveDistributor(distributor).claim(_claims);
    IBGTIncentiveDistributor(bgtIncentiveDistributor).claim(_claims);
```

Infrared: Resolved with @c09fe65efd5c...

Zenith: Verified

[I-2] Consider add make `bgtIncentiveDistributor` variable immutable and declare the variable storage in `InfraredV1_2`

SEVERITY: Informational	IMPACT: Informational
STATUS: Resolved	LIKELIHOOD: Low

Target

- InfraredV1_4

Description:

Once the `bgtIncentiveDistributor` in the initialized, there is no function to update the `bgtIncentiveDistributor`.

Recommendations:

The recommendation is make `bgtIncentiveDistributor` immutable, this immutable keywords will also prevent any storage collision.

The smart contract InfraredV1_2.sol defines all storage state.

So another recommendation is move the `bgtIncentiveDistributor` declaration to InfraredV1_2.sol to consistently manage the storage and to ensure there is no risk in storage collision.

```
    /// @notice The Infrared BGT Vault
    IInfraredVault public ibgtVault;

address immutable public bgtIncentiveDistributor;
```

Infrared: Resolved with @c09fe65efd5...

Zenith: Verified.