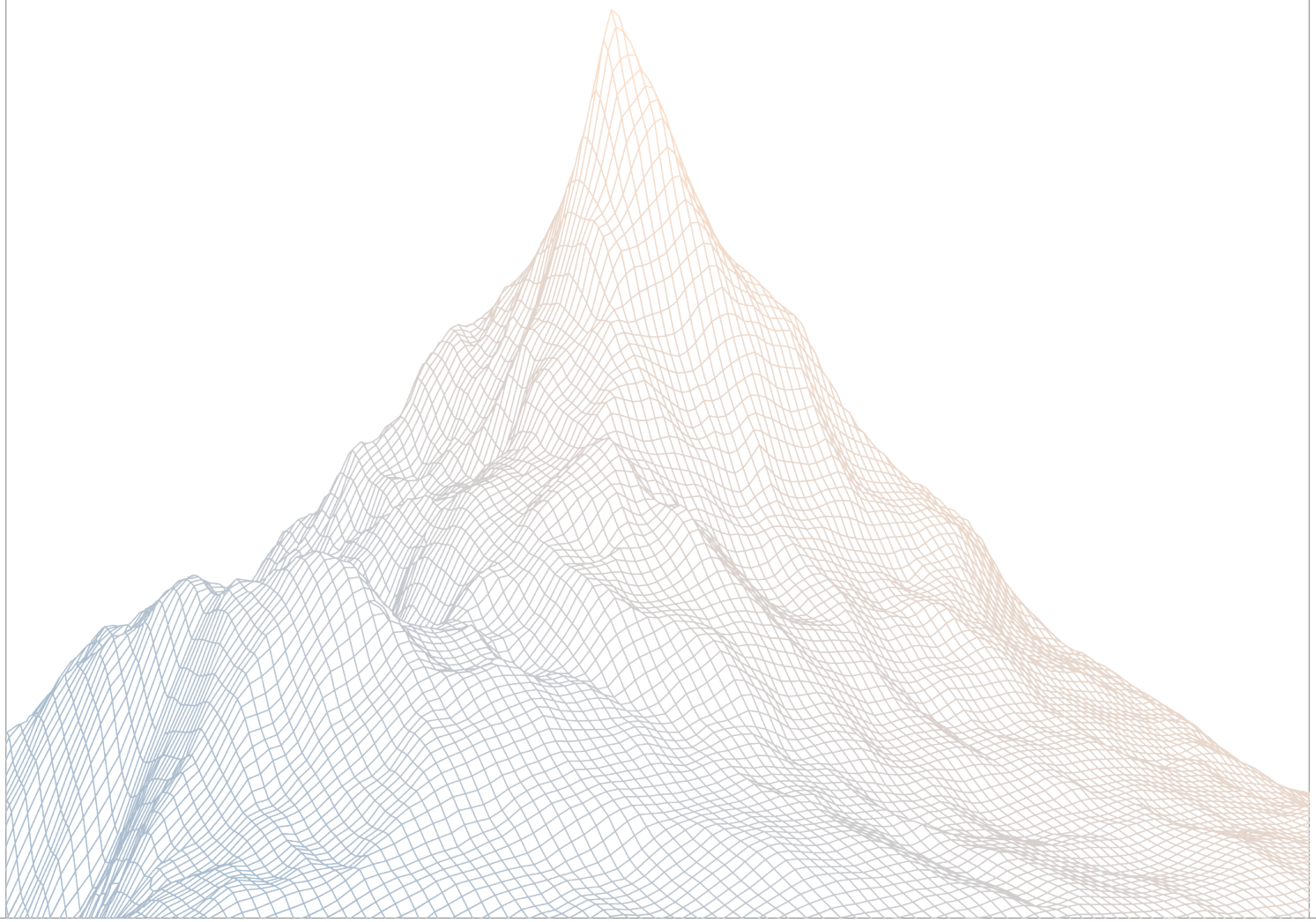
Zenith

LeverUp

Smart Contract Security Assessment

VERSION 1.1



AUDIT DATES: December 11th to December 12th, 2025

AUDITED BY: 0xAlix2
rvierdiiev

Contents

1 Introduction

1.1 About Zenith

Zenith assembles auditors with proven track records: finding critical vulnerabilities in public audit competitions.

Our audits are carried out by a curated team of the industry's top-performing security researchers, selected for your specific codebase, security needs, and budget.

Learn more about us at https://zenith.security.

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 LeverUp

LeverUp is an LP-Free perpetuals exchange delivering uncapped open interest, 100% fee redistribution to traders, and leverage up to 1001x

2.2 Scope

The engagement involved a review of the following targets:

Target	leverup-contracts-audit-zellic
Repository	https://github.com/leverup-xyz/leverup-contracts-audit-zellic
Commit Hash	`35c6a3e523bb3da667badabb4c61f55f66c46b43`
Files	`contracts/earn/LV.sol`

2.3 Audit Timeline

December 11, 2025	Audit start
December 12, 2025	Audit end
December 15, 2025	Report published

2.4 Issues Found

SEVERITY	COUNT
Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	1
Informational	2
Total Issues	**3**

3

Findings Summary

ID	Description	Status
L-1	Monad blocks in 2 days amount is calculated incorrectly	Resolved
I-1	Non-public exchange and auction availability checks	Acknowledged
I-2	Overly strict collateral check in _isLVUSDCollateralRadioLessThan100	Acknowledged

4

Findings

4.1 Low Risk

A total of 1 low risk findings were identified.

[L-1] Monad blocks in 2 days amount is calculated incorrectly

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

Target

- LV.sol

Description

Inside `updateTimeLock()` the contract checks that the new parameters do not increase by more than 2 days:

```
// Monad block time is 400ms.
// The maximum increase in 2 days is 1432000 blocks.
// This can prevent setting errors.
require(
    _enabledExchangeBlock - timeLock.enabledExchangeBlock <= 1432000,
    "EXCHANGE_BLOCK_NOT_INCREASED_MAX"
);
```

However, the stated limit is incorrect. Given that Monad block time is **400 ms**, two days contain:

```
(2 days = 172800 seconds)
172800 / 0.4s = 432000 blocks
```

Therefore the correct maximum increase should be **432,000**, not **1,432,000**. This allows the function to accept a value significantly larger than intended.

Recommendations:

Use `432000` as the correct maximum number of blocks representing 2 days on Monad.

LeverUp: Resolved with `@cd272a7bdf ...`

Zenith: Verified.

4.2 Informational

A total of 2 informational findings were identified.

[I-1] Non-public exchange and auction availability checks

SEVERITY: Informational	IMPACT: Informational
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- LV.sol

Description:

The contract exposes the time-lock struct publicly, but the actual logic determining whether exchange or auction operations are enabled is implemented only in the internal functions `_canExchange()` and `_canAuction()`.

Because these helpers are not public, they cannot be queried off-chain.

Recommendations:

Consider exposing public view wrappers (e.g., `canExchange()` and `canAuction()`).

LeverUp: Acknowledged. This will only be used to inform and protect holders, no need for complex functionality.

[I-2] Overly strict collateral check in `_isLVUSDCollateralRadioLessThan100`

SEVERITY: Informational	IMPACT: Informational
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- LV.sol

Description:

The function `_isLVUSDCollateralRadioLessThan100()` determines whether the LVUSD system is under-collateralized by checking:

```
return collateralValue < circulatingSupply;
```

This condition is extremely strict: any deficit, even a 1 wei difference, will classify the system as under-collateralized and allow `_canAuction()` to return `true`.

Recommendations:

Consider adding a small buffer to tolerate dust-level differences, or acknowledge this behavior explicitly if the auction engine already handles such thresholds.

LeverUp: Acknowledged. A cap will be set in the auction engine.