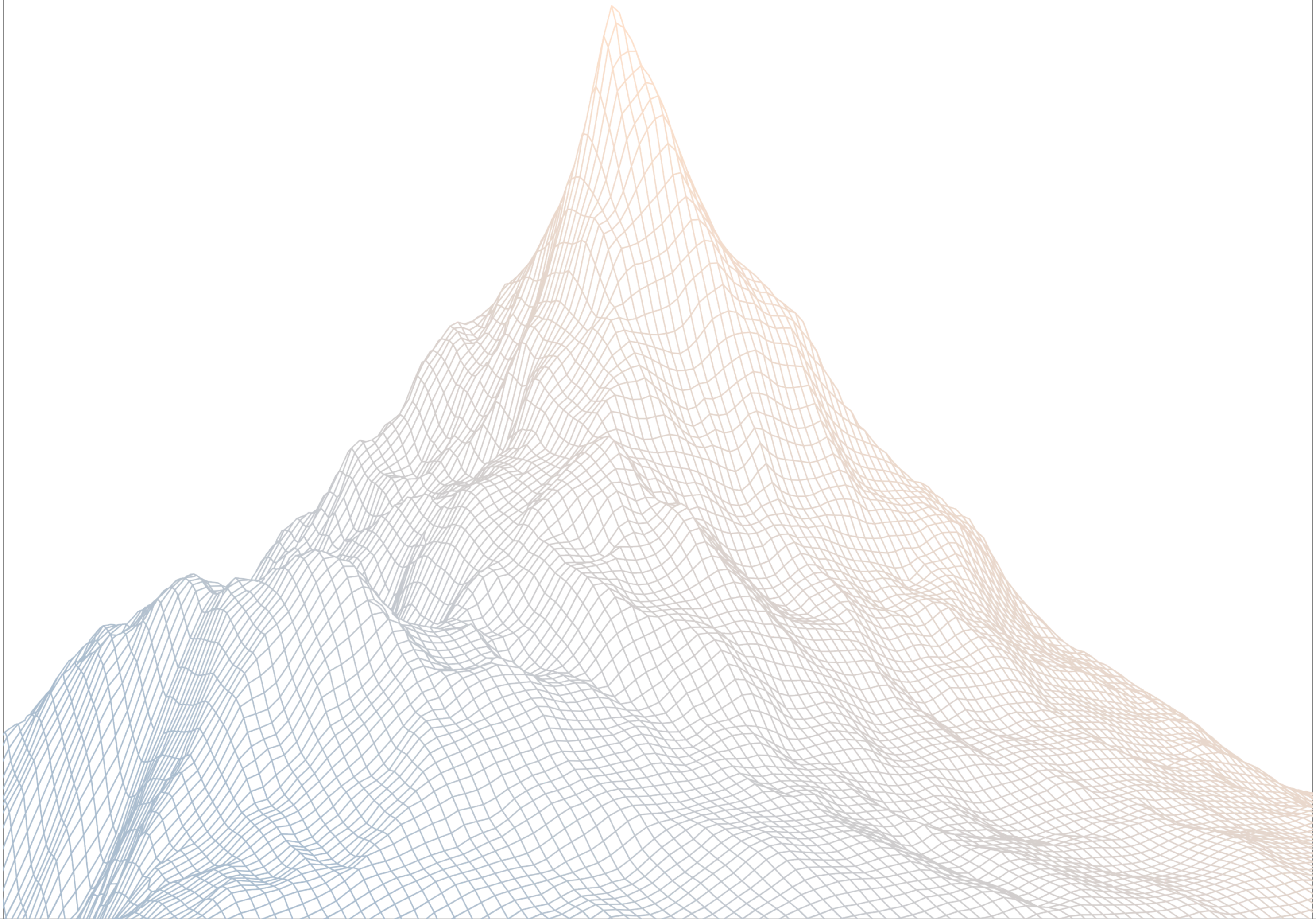
Zenith

Yala

Smart Contract Security Assessment

VERSION 1.1



AUDIT DATES:	July 15th to July 15th, 2025
AUDITED BY:	peakbolt

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 Yala

Yala is a liquidity layer for Bitcoin, unlocking Bitcoin liquidity and connecting it to cross-chain yield opportunities.

2.2 Scope

The engagement involved a review of the following targets:

Target	yala-token
Repository	https://github.com/yalaorg/yala-token
Commit Hash	`1c4eaee59863eefccaf8afeb2c41beeb0036b911`
Files	`contracts/YALA.sol`

2.3 Audit Timeline

July 15th, 2025	Audit start
July 15th, 2025	Audit end
July 15th, 2025	Report published

2.4 Issues Found

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

3 Findings Summary

ID	Description	Status
H-1	YALA contract should override ERC20 _update() instead of _beforeTokenTransfer()	Resolved
M-1	Supply of YALA OFT across all chains could exceed MAX_SUPPLY	Resolved

4 Findings

4.1 High Risk

A total of 1 high risk findings were identified.

[H-1] `YALA` contract should override ERC20 `_update()` instead of `_beforeTokenTransfer()`

SEVERITY: High	IMPACT: Medium
STATUS: Resolved	LIKELIHOOD: High

Target

- YALA.sol#L36-L46

Description:

The `YALA` OFT contract uses OZ version 5 contract but still overrides based on the OZ version 4 internal functions `_beforeTokenTransfer()`.

This prevents the compilation and deployment of `YALA` OFT contract.

```
function _beforeTokenTransfer(
    address from,
    address to,
    uint256 amount
) internal virtual override(ERC20, ERC20Pausable) {
    super._beforeTokenTransfer(from, to, amount);
}
```

Recommendations:

This can be resolved by overriding with the new `_update()` instead.

Yala: Resolved with @d9b16b080e...

Zenith: Resolved by removing ERC20Pausable so that it does not require overriding `_beforeTokenTransfer()`.

4.2 Medium Risk

A total of 1 medium risk findings were identified.

[M-1] Supply of `YALA` OFT across all chains could exceed `MAX_SUPPLY`

`SEVERITY`: Medium	`IMPACT`: Medium
`STATUS`: Resolved	`LIKELIHOOD`: Low

Target

- YALA.sol#L11-L22

Description:

The `YALA` contract relies on ERC20Capped to enforce a `MAX_SUPPLY` of 1B tokens during the deployment of the `YALA` OFT.

However, the `MAX_SUPPLY` is checked upon each deployment of the `YALA` OFT contract and that means the total supply across all chains could exceed `MAX_SUPPLY` if each deployment mints new `YALA` tokens.

```
contract YALA is OFT, ERC20Permit, ERC20Capped, ERC20Pausable {
    uint256 public constant MAX_SUPPLY = 1_000_000_000 * 1e18;

    constructor(
        string memory _name,
        string memory _symbol,
        uint256 _initialSupply,
        address _lzEndpoint,
        address _delegate,
        address _receiver,
        address _admin
    ) OFT(_name, _symbol, _lzEndpoint, _delegate) ERC20Permit(_name)
    ERC20Capped(MAX_SUPPLY) {
```

Recommendations:

Consider only minting the initial supply only on one chain and enforce the max supply there.

Yala: Resolved with @d9b16b080ec...

Zenith: Resolved by only minting initial supply on primary chain specified by a flag.