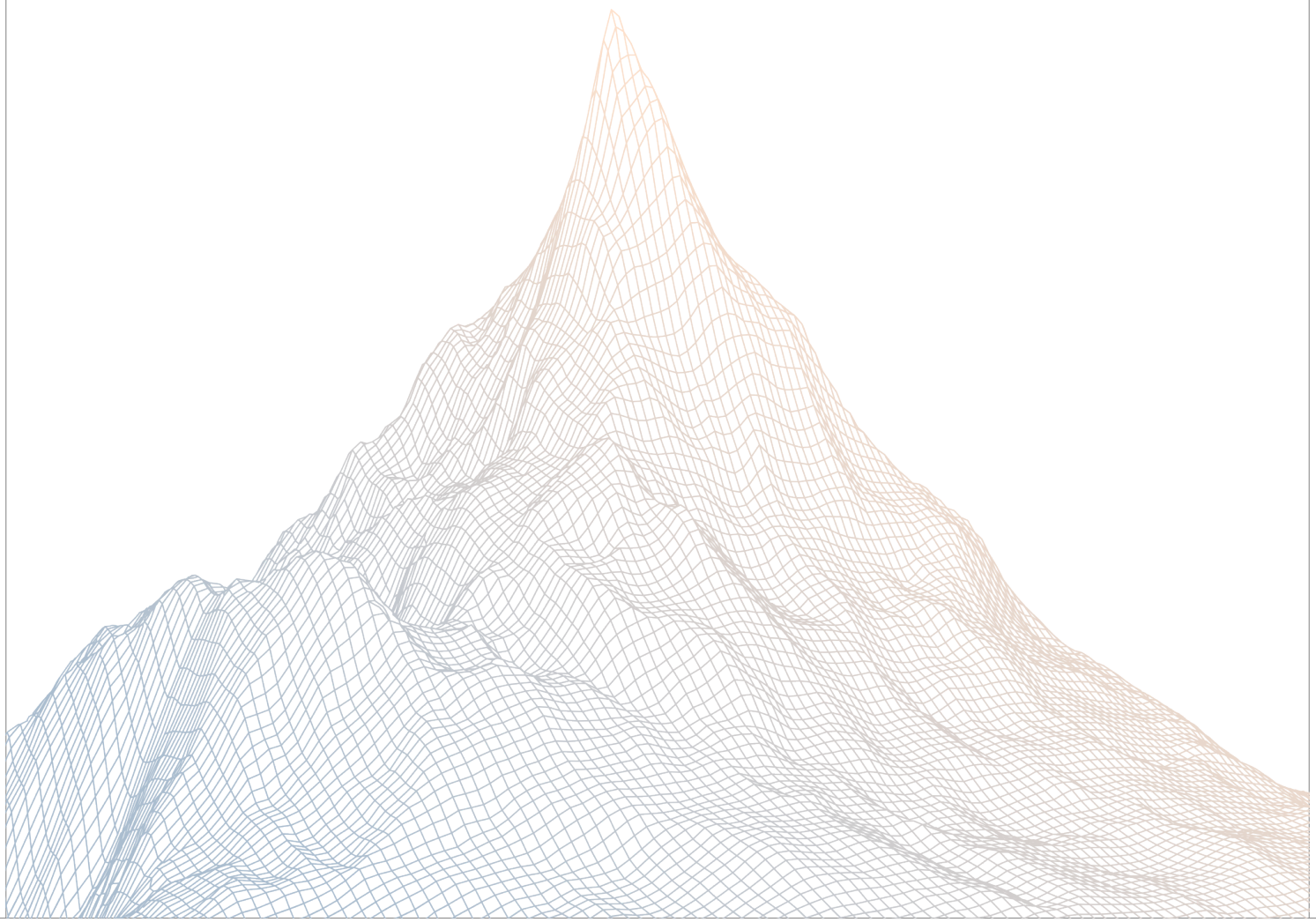
Zenith

Gondi

Smart Contract Security Assessment

VERSION 1.1



AUDIT DATES: October 17th to October 20th, 2025

AUDITED BY: SpicyMeatball
oakcobalt

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 Gondi

GONDI is a decentralized peer-to-peer non-custodial NFT lending protocol that aims to offer the most flexible and capital-efficient primitive.

GONDI V3 retains everything users love about GONDI, including pro-rata interest, instant loan refinance, and the lowest gas fees. It also introduces Tranche Seniority, enabling precise risk management. With feature upgrades and a streamlined user experience, GONDI V3 is the most advanced and efficient NFT lending protocol.

2.2 Scope

The engagement involved a review of the following targets:

Target	florida-contracts
Repository	https://github.com/pixeldaogg/florida-contracts
Commit Hash	`7ba25f98633d7cccdcb59e1f6cdd4823de8bbdcd`
Files	`Diff from dab0636a4d55901df7e74e27363399d666af6a72`

2.3 Audit Timeline

October 17, 2025	Audit start
October 20, 2025	Audit end
November 7, 2025	Report published

2.4 Issues Found

SEVERITY	COUNT
Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	3
Informational	1
Total Issues	**4**

3

Findings Summary

ID	Description	Status
L-1	Incomplete Big Blocks Preparation	Acknowledged
L-2	setProtocolFee allow event spamming and stale pending protocol fee states	Acknowledged
L-3	WrappedCryptopunksStub are missing methods called from PurchaseBundler	Acknowledged
I-1	Protocol Fee Update Notice Should Use Immutable Pattern for Deployment Flexibility	Acknowledged

4

Findings

4.1 Low Risk

A total of 3 low risk findings were identified.

[L-1] Incomplete Big Blocks Preparation

SEVERITY: Low	IMPACT: Low
STATUS: Acknowledged	LIKELIHOOD: High

Target

- big_blocks_hyperevm.py#L42-L50

Description:

The deploy/hyperliquid/big_blocks_hyperevm.py script contains multiple issues that will prevent successful execution of the big blocks activation process.

1. Missing Resp Variable

```
print("Enable big blocks action response:", resp)  #  'resp' is not defined
```

2. Commented Out Nonce

```
# nonce = exchange._get_nonce(account.address)  #  This is commented out
```

3. Incomplete Action - Only signed, but never broadcasted to the network

```
# --- Sign the action ---
timestamp = get_timestamp_ms()

sig = sign_l1_action(account, action, None, timestamp,
    exchange.expires_after, True)
```

Recommendations

Fix the inconsistency and add the missing API call (e.g. `requests.post()`) to enable big blocks for the account

Gondi: Acknowledged. Will address in future changes.

[L-2] setProtocolFee allow event spamming and stale pending protocol fee states

SEVERITY: Low	IMPACT: Low
STATUS: Acknowledged	LIKELIHOOD: Medium

Target

- WithProtocolFee.sol#L73-L74

Description:

The `setProtocolFee()` function in WithProtocolFee.sol does not update `_pendingProtocolFeeSetTime` and clear the pending state `_pendingProtocolFee` after updating the protocol fee. This creates several issues:

1. Event spamming:

Since `setProtocolFee()` is permissionless, anyone can still call `setProtocolFee()` which emit multiple `ProtocolFeeUpdated` events. Event spamming may cause off-chain monitoring to misinterpret duplicated events or stale pending state.

2. State inconsistency:

After execution, the contract still shows a "pending" fee even though it has been applied.

Recommendations:

Reset both pending state variables after fee update. For example,

```
function setProtocolFee() external virtual {
    if (block.timestamp < _pendingProtocolFeeSetTime + FEE_UPDATE_NOTICE) {
        revert TooSoonError();
    }
    ProtocolFee memory protocolFee = _pendingProtocolFee;
    _protocolFee = protocolFee;

    // Clear pending state to prevent spam and maintain consistency
    _pendingProtocolFee = ProtocolFee(address(0), 0);
```

```
    _pendingProtocolFeeSetTime = type(uint256).max;

    emit ProtocolFeeUpdated(protocolFee);
}
```

Gondi: Acknowledged. Will address in future changes.

[L-3] WrappedCryptopunksStub are missing methods called from PurchaseBundler

SEVERITY: Low	IMPACT: Low
STATUS: Acknowledged	LIKELIHOOD: Medium

Target

- WrappedCryptopunksStub.sol#L4

Description:

The WrappedCryptopunksStub contract is missing methods that are called in production code, leading to inconsistent error handling and potential function selector failures. The missing methods are: `mint(uint256)` , `burn(uint256)` and `transferFrom(address,address,uint256)`. These are called from PurchaseBundler.sol. When these methods are called on the stub contract from PurchaseBundler.sol, the intended error message "Not implemented" will not be returned. This breaks the consistent error-handling pattern established by the stub's design.

Impact: Low - While this doesn't break core protocol functionality, it creates inconsistent error handling.

Recommendations:

Add the missing methods to maintain consistent error handling:

Gondi: Acknowledged. Will address in future changes.

4.2 Informational

A total of 1 informational findings were identified.

[I-1] Protocol Fee Update Notice Should Use Immutable Pattern for Deployment Flexibility

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

Target

- WithProtocolFee.sol#L17

Description:

The `FEE_UPDATE_NOTICE` constant in WithProtocolFee.sol is hardcoded to 7 days, which was recently changed from 30 days in the previous commit. This approach creates limitations:

1. Deployment Inflexibility: The same contract has to be rewritten when changing a configurable parameter. Future adjustment based on chain or community expectations would require rewriting the contract.
2. Inconsistent Pattern: Unlike `TwoStepOwned.sol`, which uses `immutable` for `MIN_WAIT_TIME`, this contract uses a `constant`.

```
uint256 public constant FEE_UPDATE_NOTICE = 7 days;
```

Recommendations:

Change FEE_UPDATE_NOTICE from `constant` to `immutable` to allow deployment-time configuration. This maintains the same security guarantees as `constant` while avoiding contract rewrite.

Gondi: Acknowledged. Will address in future changes.