LTV protocol

LTV TRANSPARENT UPGRADEABLE BEACON PROXY SMART CONTRACTS SECURITY AUDIT REPORT

OXORIO

NOVEMBER 10, 2025

1 EXECUTIVE SUMMARY

OXORIO

1.1 EXECUTIVE SUMMARY

This document presents the smart contracts security audit conducted by Oxorio for LTV Protocol's Transparent Upgradeable Beacon Proxy.

The LTV Protocol is a revolutionary Curatorless Leveraged Tokenized Vault that maintains a constant target Loan-To-Value (LTV) ratio without requiring a central curator. Built on the foundation of two interconnected EIP-4626 vaults, it enables users to deposit and withdraw funds while receiving tokenized shares representing their leveraged positions. The protocol's core innovation lies in its auction-based stimulus system that incentivizes users to participate in rebalancing actions through rewards or fees. This mechanism ensures alignment with the target LTV while providing basic MEV protection against frontrunning.

Transparent Upgradeable Beacon Proxy is a Solidity-based proxy mechanism that combines the beacon pattern with transparent upgradeability. The beacon allows multiple proxy instances to share a single implementation, while the transparent upgrade feature enables updating an individual proxy's logic when needed, without affecting other contracts connected to the same beacon.

The audit process involved a comprehensive approach, including manual code review, automated analysis, and extensive testing and simulations of the smart contracts to assess the project's security and functionality. The audit covered a total of 3 smart contracts, encompassing 51 lines of code. The codebase was thoroughly examined, with the audit team collaborating closely with LTV Protocol. For an in-depth explanation of used the smart contract security audit methodology, please refer to the Security Assessment Methodology section of this document.

Throughout the audit, a collaborative approach was maintained with LTV Protocol to address all concerns identified within the audit's scope. Each issue has been either resolved or formally acknowledged by LTV Protocol, contributing to the robustness of the project.

As a result, following a comprehensive review, our auditors have verified that the Transparent Upgradeable Beacon Proxy, as of audited commit 22aca3f1d1cd76999f3c7775ca606cd0a80ac9ab, has met the security and functionality requirements established for this audit, based on the code and documentation provided, and operates as intended within the defined scope.

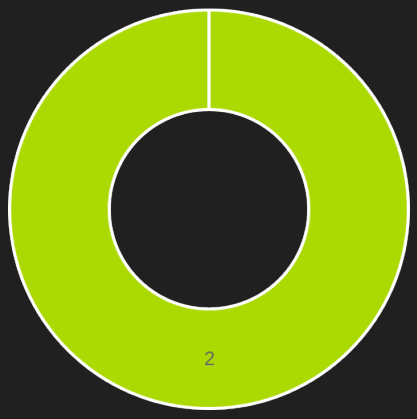
1.2 SUMMARY OF FINDINGS

The table below provides a comprehensive summary of the audit findings, categorizing each by status and severity level. For a detailed description of the severity levels and statuses of findings, see the Findings Classification Reference section.

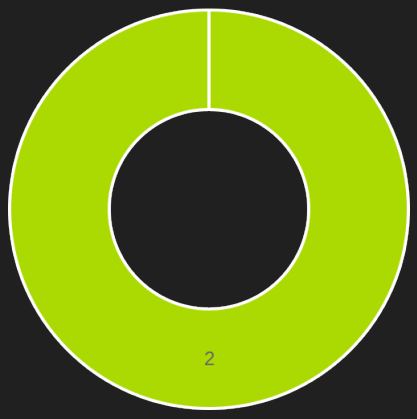
Detailed technical information on the audit findings, along with our recommendations for addressing them, is provided in the Findings Report section for further reference.

All identified issues have been addressed, with LTV Protocol fixing them or formally acknowledging their status.

Severity	TOTAL	NEW	FIXED	ACKNOWLEDGED	NO ISSUE
CRITICAL	**0**	0	0	0	0
MAJOR	**0**	0	0	0	0
WARNING	**0**	0	0	0	0
INFO	**2**	0	2	0	0
TOTAL	**2**	**0**	**2**	**0**	**0**



Issue distribution by severity



Issue distribution by status

2 AUDIT OVERVIEW

CONTENTS

2.1 DISCLAIMER

At the request of the client, Oxorio consents to the public release of this audit report. The information contained herein is provided "as is" without any representations or warranties of any kind. Oxorio disclaims all liability for any damages arising from or related to the use of this audit report. Oxorio retains copyright over the contents of this report.

This report is based on the scope of materials and documentation provided to Oxorio for the security audit as detailed in the Executive Summary and Audited Files sections. The findings presented in this report may not encompass all potential vulnerabilities. Oxorio delivers this report and its findings on an as-is basis, and any reliance on this report is undertaken at the user's sole risk. It is important to recognize that blockchain technology remains in a developmental stage and is subject to inherent risks and flaws.

This audit does not extend beyond the programming language of smart contracts to include areas such as the compiler layer or other components that may introduce security risks. Consequently, this report should not be interpreted as an endorsement of any project or team, nor does it guarantee the security of the project under review.

THE CONTENT OF THIS REPORT, INCLUDING ITS ACCESS AND/OR USE, AS WELL AS ANY ASSOCIATED SERVICES OR MATERIALS, MUST NOT BE CONSIDERED OR RELIED UPON AS FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER PROFESSIONAL ADVICE. Third parties should not rely on this report for making any decisions, including the purchase or sale of any product, service, or asset. Oxorio expressly disclaims any liability related to the report, its contents, and any associated services, including, but not limited to, implied warranties of merchantability, fitness for a particular purpose, and non-infringement. Oxorio does not warrant, endorse, or take responsibility for any product or service referenced or linked within this report.

For any decisions related to financial, legal, regulatory, or other professional advice, users are strongly encouraged to consult with qualified professionals.

2.2 PROJECT BRIEF

Title	Description
Client	LTV Protocol
Project name	LTV Transparent Upgradeable Beacon Proxy
Category	Upgradeable Proxy Infrastructure
Website	https://ltv.finance
Documentation	README, LTV_Curatorless_Leveraged_Tokenized_Vault_with_a_Constant_Target_Loan-To-Value_Ratio.pdf
Initial Commit	a047cab39328652d37cac19cb65656ba89094269
Final Commit	22aca3f1d1cd76999f3c7775ca606cd0a80ac9ab
Platform	L1
Network	Ethereum
Languages	Solidity
Lead Auditor	Artem Belozerov - artem@oxor.io
Project Manager	Elena Kozmiryuk - elena@oxor.io

2.3 PROJECT TIMELINE

The key events and milestones of the project are outlined below.

Date	Event
November 3, 2025	Client engaged Oxorio requesting an audit.
November 7, 2025	The audit team initiated work on the project.
November 7, 2025	Submission of the initial audit report.
November 9, 2025	Client's feedback on the initial audit report was received.
November 10, 1970	The audit team commenced work on a re-audit of the project.
November 10, 1970	Submission of the final audit report incorporating client's verified fixes.

2.4 AUDITED FILES

The following table contains a list of the audited files. The scc tool was used to count the number of lines and assess complexity of the files.

	File	Lines	Blanks	Comments	Code	Complexity
1	src/BeaconProxyAdmin.sol	17	3	1	**13**	0%
2	src/interfaces/ITransparentUpgradeableBeaconProxy.sol	9	2	2	**5**	0%
3	src/TransparentUpgradeableBeaconProxy.sol	43	8	2	**33**	18%
	Total	**69**	**13**	**5**	**51**	**12%**

Lines: The total number of lines in each file. This provides a quick overview of the file size and its contents.

Blanks: The count of blank lines in the file.

Comments: This column shows the number of lines that are comments.

Code: The count of lines that actually contain executable code. This metric is essential for understanding how much of the file is dedicated to operational elements rather than comments or whitespace.

Complexity: This column shows the file complexity per line of code. It is calculated by dividing the file's total complexity (an approximation of cyclomatic complexity that estimates logical depth and decision points like loops and conditional branches) by the number of executable lines of code. A higher value suggests greater complexity per line, indicating areas with concentrated logic.

2.5 PROJECT OVERVIEW

The project provides a Solidity-based proxy architecture that combines the beacon upgrade pattern with the transparent proxy access control model. The core idea of the design is to enable multiple proxy instances to share a single beacon contract that defines the implementation logic, while still allowing individual proxies to be upgraded independently when needed.

In the beacon model, the implementation address is stored in a dedicated Beacon contract, and any proxy that references it will delegate calls to the implementation retrieved from the beacon. This allows multiple proxy deployments to operate with a common logic layer and makes version updates efficient.

The “transparent” component ensures that administrative upgrade operations are kept separate from normal user interactions. Calls from non-admin addresses are forwarded to the implementation contract, while calls from the admin behave differently and are used to manage upgrade operations. This prevents accidental or unauthorized activation of administrative functionality by regular users or external integrations.

Overall, the project provides a flexible mechanism for upgradeable smart contract deployments that can scale across multiple instances while maintaining clear separation of roles and consistent upgrade behavior.

2.6 CODEBASE QUALITY ASSESSMENT

The Codebase Quality Assessment table offers a comprehensive assessment of various code metrics, as evaluated by our team during the audit, to gauge the overall quality and maturity of the project's codebase. By evaluating factors such as complexity, documentation and testing coverage to best practices, this table highlights areas where the project excels and identifies potential improvement opportunities. Each metric receives an individual rating, offering a clear snapshot of the project's current state, guiding prioritization for refactoring efforts, and providing insights into its maintainability, security, and scalability. For a detailed description of the categories and ratings, see the Codebase Quality Assessment Reference section.

Category	Assessment	Result
Access Control	The ownable beacon proxy pattern distributes responsibility between the proxy admin and the beacon owner. In this project, this behavior is intentional and follows the design purpose of enabling independent upgrade authority.	**Excellent**
Arithmetic	The contracts do not perform any complex arithmetic operations, and arithmetic safety is not a concern in this context.	**Not Applicable**
Complexity	The architecture is straightforward, and the relationships between the proxy, beacon, and implementation contracts are easy to follow.	**Excellent**
Data Validation	Input validation mechanics are appropriate for the intended contract interactions, and no unsafe inputs affecting upgrade or delegation behavior were identified.	**Excellent**
Decentralization	Decentralization considerations are not applicable in this context, as the system is intentionally designed to rely on upgrade authority roles.	**Not Applicable**
Documentation	The contracts are well-structured and understandable. To provide even greater clarity into the upgrade processes, we recommend adding more detailed, function-level specifications.	**Excellent**
External Dependencies	The contracts do not integrate external protocols or third-party systems beyond standard low-level Solidity behavior and proxy standards.	**Not Applicable**

Category	Assessment	Result
Error Handling	Errors are handled consistently using `revert` and `require`, and state changes are gated appropriately to prevent unintended behavior.	**Excellent**
Logging and Monitoring	Event emission behavior is appropriate for upgrade-related operations, and no additional monitoring mechanisms are required.	**Excellent**
Low-Level Calls	Delegate calls and low-level calls are limited to proxy forwarding behavior and are implemented according to established proxy standards.	**Excellent**
Testing and Verification	Test coverage is sufficient for core functionality, and expected behavior of proxy upgrade interactions is verified.	**Excellent**

2.7 FINDINGS BREAKDOWN BY FILE

This table provides an overview of the findings across the audited files, categorized by severity level. It serves as a useful tool for identifying areas that may require attention, helping to prioritize remediation efforts, and provides a clear summary of the audit results.

File	TOTAL	CRITICAL	MAJOR	WARNING	INFO
src/BeaconProxyAdmin.sol	**1**	0	0	0	1
src/TransparentUpgradeableBeaconProxy.sol	**1**	0	0	0	1
src/interfaces/ITransparentUpgradeableBeaconProxy.sol	**1**	0	0	0	1

2.8 CONCLUSION

A comprehensive audit was conducted on 3 smart contracts, revealing 2 informational observations related to role responsibility clarity and parameter naming consistency. The audit confirmed that the contracts correctly implement the intended transparent upgradeable beacon proxy pattern and operate as designed. However, the review noted that the separation of privileges between the proxy admin and the beacon owner, while intentional, may be non-obvious and should be clearly documented to prevent misunderstanding during system operation and maintenance.

The proposed changes are aimed at improving documentation clarity regarding role responsibilities and the meaning of key function parameters, helping to prevent ambiguity when performing upgrades. These recommendations are based on industry best practices and are intended to enhance maintainability and ensure proper operational transparency. We strongly advise addressing the identified informational findings to avoid potential misinterpretation and reinforce the long-term reliability of the system.

A comprehensive audit was conducted on 3 smart contracts. No critical or major issues were identified, and only 2 informational notes were reported. The audit highlighted several behavioral and operational nuances, primarily related to role-based access control boundaries and clarity of function parameter semantics.

Following our initial audit, LTV Protocol worked closely with our team to address the identified issues.
We recommend clarifying role responsibilities within the proxy–beacon ownership model and improving the specificity of parameter naming and documentation to avoid ambiguity in privilege interpretation. All identified issues have been successfully addressed or formally acknowledged.

As a result, the project has passed our audit. Our auditors have verified that the project, as of audited commit `22aca3f1d1cd76999f3c7775ca606cd0a80ac9ab`, operates as intended within the defined scope, based on the information and code provided at the time of evaluation. The robustness of the codebase has been improved, meeting the necessary security and functionality requirements established for this audit.

To enhance the project's security and production readiness, we recommend conducting periodic reviews following any contract upgrades or architectural changes, as well as maintaining adequate test coverage to ensure continued reliability over time.

3 FINDINGS REPORT

OXORIO

3.1 CRITICAL

No issues found

3.2 MAJOR

No issues found

3.3 WARNING

No issues found

3.4 INFO

I-01	Mixing responsibilities of the proxy admin and the beacon owner in `TransparentUpgradeableBeaconProxy`
Severity	**INFO**
Status	• FIXED

Location

File	Location	Line
TransparentUpgradeableBeaconProxy.sol	contract `TransparentUpgradeableBeaconProxy` > `constructor`	18

Description

In the `TransparentUpgradeableBeaconProxy` contract, the admin role is granted privileges to update the beacon address by calling the `upgradeBeaconToAndCall` function. This call overwrites the beacon address stored in the hard-coded `BEACON_SLOT`:

```
bytes32 internal constant BEACON_SLOT =
0xa3f0ad74e5423aebfd80d3ef4346578335a9a72aeaee59ff6cb3582b35133d50;
```

At the same time, the beacon contract itself has an owner role, which is separate from the proxy admin. The beacon owner has the ability to update the final logic implementation. Therefore, the beacon owner is able to provide an implementation that, when executed, can modify the proxy's state, including the `BEACON_SLOT`, for example:

```
bytes32 constant BEACON_SLOT =
0xa3f0ad74e5423aebfd80d3ef4346578335a9a72aeaee59ff6cb3582b35133d50
assembly {
    sstore(BEACON_SLOT, newBeaconAddress)
}
```

As a result, the beacon owner effectively gains the ability to change the beacon address stored in the proxy, which corresponds to privileges of the proxy admin.

Although this behavior aligns with the intended ownable beacon proxy design, it may not be immediately evident when distributing responsibilities between these roles.

Recommendation

We recommend explicitly documenting the capabilities of both roles, the proxy admin and the beacon owner, to avoid ambiguity in understanding the boundaries of their privileges.

Update

Client's response

Fixed at 265d1475530965e102cf4eba9f6736021b09a809.

I-02	Ambiguous naming of function parameters in `BeaconProxyAdmin`, `ITransparentUpgradeableBeaconProxy`
Severity	**INFO**
Status	• FIXED

Location

File	Location	Line
BeaconProxyAdmin.sol	contract `BeaconProxyAdmin` > function `upgradeBeaconToAndCall`	12
ITransparentUpgradeableBeaconProxy.sol	interface `ITransparentUpgradeableBeaconProxy` > function `upgradeBeaconToAndCall`	7

Description

In the mentioned locations, the function signature for `upgradeBeaconToAndCall` uses a parameter named `implementation`, while the actual function calls pass a `beacon` address instead:

```
constructor(address beacon, address initialOwner, bytes memory data) payable {
    ERC1967Utils.upgradeBeaconToAndCall(beacon, data);
    // ...

function _dispatchUpgradeBeaconToAndCall() private {
    (address newBeacon, bytes memory data) = abi.decode(msg.data[4:], (address, bytes));
    ERC1967Utils.upgradeBeaconToAndCall(newBeacon, data);
    // ...
```

Recommendation

We recommend clearly specifying in the specification the meaning and purpose of the function parameters to prevent ambiguity between the concept of a logic implementation contract and the beacon contract.

Update

Client's response

Fixed at 273f2ea4c2f0ff7d5c5fd7da8bb3c933d930535d.

4. APPENDIX

4.1 SECURITY ASSESSMENT METHODOLOGY

Oxorio's smart contract security audit methodology is designed to ensure the security, reliability, and compliance of smart contracts throughout their development lifecycle. Our process integrates the Smart Contract Security Verification Standard (SCSVS) with our advanced techniques to address complex security challenges. For a detailed look at our approach, please refer to the full version of our methodology. Here is a concise overview of our auditing process:

1. Project Architecture Review

All necessary information about the smart contract is gathered, including its intended functionality and dependencies. This stage sets the foundation by reviewing documentation, business logic, and initial code analysis.

2. Vulnerability Assessment

This phase involves a deep dive into the smart contract's code to identify security vulnerabilities. Rigorous testing and review processes are applied to ensure robustness against potential attacks.

This stage is focused on identifying specific vulnerabilities within the smart contract code. It involves scanning and testing the code for known security weaknesses and patterns that could potentially be exploited by malicious actors.

3. Security Model Evaluation

The smart contract’s architecture is assessed to ensure it aligns with security best practices and does not introduce potential vulnerabilities. This includes reviewing how the contract integrates with external systems, its compliance with security best practices, and whether the overall design supports a secure operational environment.

This phase involves a analysis of the project's documentation, the consistency of business logic as documented versus implemented in the code, and any assumptions made during the design and development phases. It assesses if the contract's architectural design adequately addresses potential threats and integrates necessary security controls.

4. Cross-Verification by Multiple Auditors

Typically, the project is assessed by multiple auditors to ensure a diverse range of insights and thorough coverage. Findings from individual auditors are cross-checked to verify accuracy and completeness.

5. Report Consolidation

Findings from all auditors are consolidated into a single, comprehensive audit report. This report outlines potential vulnerabilities, areas for improvement, and an overall assessment of the smart contract's security posture.

6. Reaudit of Revised Submissions

Post-review modifications made by the client are reassessed to ensure that all previously identified issues have been adequately addressed. This stage helps validate the effectiveness of the fixes applied.

7. Final Audit Report Publication

The final version of the audit report is delivered to the client and published on Oxorio's official website. This report includes detailed findings, recommendations for improvement, and an executive summary of the smart contract's security status.

4.2 CODEBASE QUALITY ASSESSMENT REFERENCE

The tables below describe the codebase quality assessment categories and rating criteria used in this report.

Category	Description
Access Control	Evaluates the effectiveness of mechanisms controlling access to ensure only authorized entities can execute specific actions, critical for maintaining system integrity and preventing unauthorized use.
Arithmetic	Focuses on the correct implementation of arithmetic operations to prevent vulnerabilities like overflows and underflows, ensuring that mathematical operations are both logically and semantically accurate.
Complexity	Assesses code organization and function clarity to confirm that functions and modules are organized for ease of understanding and maintenance, thereby reducing unnecessary complexity and enhancing readability.
Data Validation	Assesses the robustness of input validation to prevent common vulnerabilities like overflow, invalid addresses, and other malicious input exploits.
Decentralization	Reviews the implementation of decentralized governance structures to mitigate insider threats and ensure effective risk management during contract upgrades.
Documentation	Reviews the comprehensiveness and clarity of code documentation to ensure that it provides adequate guidance for understanding, maintaining, and securely operating the codebase.
External Dependencies	Evaluates the extent to which the codebase depends on external protocols, oracles, or services. It identifies risks posed by these dependencies, such as compromised data integrity, cascading failures, or reliance on centralized entities. The assessment checks if these external integrations have appropriate fallback mechanisms or redundancy to mitigate risks and protect the protocol's functionality.
Error Handling	Reviews the methods used to handle exceptions and errors, ensuring that failures are managed gracefully and securely.
Logging and Monitoring	Evaluates the use of event auditing and logging to ensure effective tracking of critical system interactions and detect potential anomalies.
Low-Level Calls	Reviews the use of low-level constructs like inline assembly, raw `call` or `delegatecall`, ensuring they are justified, carefully implemented, and do not compromise contract security.

Category	Description
Testing and Verification	Reviews the implementation of unit tests and integration tests to verify that codebase has comprehensive test coverage and reliable mechanisms to catch potential issues.

4.2.1 Rating Criteria

Rating	Description
Excellent	The system is flawless and surpasses standard industry best practices.
Good	Only minor issues were detected; overall, the system adheres to established best practices.
Fair	Issues were identified that could potentially compromise system integrity.
Poor	Numerous issues were identified that compromise system integrity.
Absent	A critical component is absent, severely compromising system safety.
Not Applicable	This category does not apply to the current evaluation.

4.3 FINDINGS CLASSIFICATION REFERENCE

4.3.1 Severity Level Reference

The following severity levels were assigned to the issues described in the report:

Title	Description
CRITICAL	Issues that pose immediate and significant risks, potentially leading to asset theft, inaccessible funds, unauthorized transactions, or other substantial financial losses. These vulnerabilities represent serious flaws that could be exploited to compromise or control the entire contract. They require immediate attention and remediation to secure the system and prevent further exploitation.
MAJOR	Issues that could cause a significant failure in the contract's functionality, potentially necessitating manual intervention to modify or replace the contract. These vulnerabilities may result in data corruption, malfunctioning logic, or prolonged downtime, requiring substantial operational changes to restore normal performance. While these issues do not immediately lead to financial losses, they compromise the reliability and security of the contract, demanding prioritized attention and remediation.
WARNING	Issues that might disrupt the contract's intended logic, affecting its correct functioning or making it vulnerable to Denial of Service (DDoS) attacks. These problems may result in the unintended triggering of conditions, edge cases, or interactions that could degrade the user experience or impede specific operations. While they do not pose immediate critical risks, they could impact contract reliability and require attention to prevent future vulnerabilities or disruptions.
INFO	Issues that do not impact the security of the project but are reported to the client's team for improvement. They include recommendations related to code quality, gas optimization, and other minor adjustments that could enhance the project's overall performance and maintainability.

4.3.2 Status Level Reference

Based on the feedback received from the client's team regarding the list of findings discovered by the contractor, the following statuses were assigned to the findings:

Title	Description
NEW	Waiting for the project team's feedback.

Title	Description
FIXED	Recommended fixes have been applied to the project code and the identified issue no longer affects the project's security.
ACKNOWLEDGED	The project team is aware of this finding and acknowledges the associated risks. This finding may affect the overall security of the project; however, based on the risk assessment, the team will decide whether to address it or leave it unchanged.
NO ISSUE	Finding does not affect the overall security of the project and does not violate the logic of its work.

4.4 ABOUT OXORIO

OXORIO is a blockchain security firm that specializes in smart contracts, zk-SNARK solutions, and security consulting. With a decade of blockchain development and five years in smart contract auditing, our expert team delivers premier security services for projects at any stage of maturity and development.

Since 2021, we've conducted key security audits for notable DeFi projects like Lido, 1Inch, Rarible, and deBridge, prioritizing excellence and long-term client relationships. Our co-founders, recognized by the Ethereum and Web3 Foundations, lead our continuous research to address new threats in the blockchain industry. Committed to the industry's trust and advancement, we contribute significantly to security standards and practices through our research and education work.

Our contacts:

- oxor.io
- ping@oxor.io
- Github
- Linkedin
- Twitter

THANK YOU FOR CHOOSING