CODE DEVELOPED BY

BGD Labs

AAVE V3.3.0 SMART CONTRACTS SECURITY AUDIT REPORT

OXORIO

JANUARY 29, 2025

1 EXECUTIVE SUMMARY

OXORIO

1.1 EXECUTIVE SUMMARY

This document presents the smart contracts security audit conducted by Oxorio for BGD Labs's Aave v3.3.0 Update.

BGD Labs (Bored Ghosts Developing Labs) is a team of technical contributors founded by three core members of the Aave community with extensive expertise in Aave and decentralized finance. Their mission is to provide core development support for the Aave ecosystem, ensuring the security and continuous innovation of the protocol while maintaining an open and collaborative approach. BGD Labs is dedicated exclusively to Aave, focusing on creating open-source solutions for the benefit of the Aave DAO and its community.

The Aave Protocol v3.3.0 introduces several important upgrades to enhance functionality and prepare for future integrations. Firstly, it introduces bad debt accounting, a key feature designed to ensure compatibility with the upcoming Umbrella system, which aims to improve the protocol's overall debt management and risk mitigation. Secondly, the liquidations mechanism has been improved, specifically with refinements to the close factor function, providing greater stability and efficiency during liquidation events while including complementary mechanisms to further optimize this process. Additionally, the update optimizes ReserveConfiguration bit operations, enhancing operational efficiency and reducing gas costs related to managing reserves

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 a changes in total of 10 smart contracts, encompassing 2905 lines of code. The codebase was thoroughly examined, with the audit team collaborating closely with BGD Labs and referencing the provided documentation to address any questions regarding the expected behavior. 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 BGD Labs to address all concerns identified within the audit's scope. Each issue has been either resolved or formally acknowledged by BGD Labs, contributing to the robustness of the project.

As a result, following a comprehensive review, our auditors have verified that the Aave Protocol v3.3.0, as of audited commit `21c30148d1484ddec57f5d223f530179b103cae6`, 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.

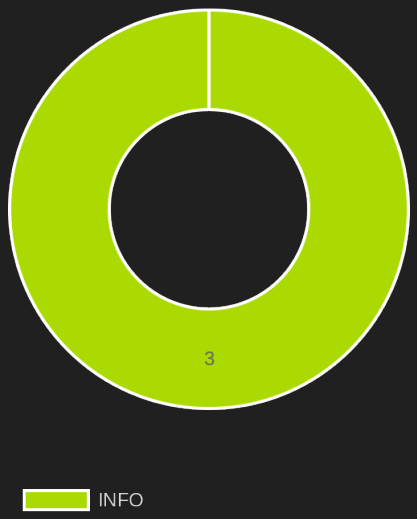
1.2 SUMMARY OF FINDINGS

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.

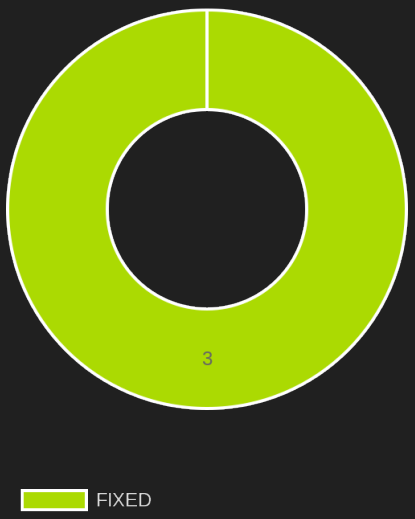
Detailed technical information on the audit findings, along with our recommendations for addressing them, is provided in the Findings Report section for further reference.

All identified issues have been addressed, with BGD Labs 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	**3**	0	3	0	0
TOTAL	**3**	**0**	**3**	**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	Aave DAO
Code developed by	BGD Labs
Project name	Aave v3.3.0
Category	Lending
Website	aave.com
Repository	github.com/aave-dao/aave-v3-origin/pull/87/
Documentation	github.com/aave-dao/aave-v3-origin/pull/87/files#diff-4d78933bb89fd6e8ba57d10835beac16cd962b99e890e90b95105a85836f0d74
Initial Commit	`f4a12ecf6754e7aa654492550d7137871bfe8a4d`
Final Commit	`21c30148d1484ddec57f5d223f530179b103cae6`
Platform	L1, L2
Languages	Solidity
Lead Auditor	Alexander Mazaletskiy - am@oxor.io
Project Manager	Natalia Demidova - nataly@oxor.io

2.3 PROJECT TIMELINE

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

Date	Event
January 7, 2025	Client engaged Oxorio requesting an audit.
January 15, 2025	The audit team initiated work on the project.
January 29, 2025	Submission of the comprehensive audit report.

2.4 AUDITED FILES

The following table contains a list of the audited files. An audit of all changes in these files starting from commit `f4a12ecf6754e7aa654492550d7137871bfe8a4d` was conducted. 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/contracts/protocol/libraries/configuration/ReserveConfiguration.sol	587	57	228	**302**	4
2	src/contracts/protocol/libraries/helpers/Errors.sol	107	1	6	**100**	0
3	src/contracts/protocol/libraries/logic/BorrowLogic.sol	235	27	31	**177**	11
4	src/contracts/protocol/libraries/logic/BridgeLogic.sol	155	23	32	**100**	10
5	src/contracts/protocol/libraries/logic/ConfiguratorLogic.sol	215	28	39	**148**	1
6	src/contracts/protocol/libraries/logic/LiquidationLogic.sol	726	68	160	**498**	11
7	src/contracts/protocol/libraries/logic/ReserveLogic.sol	314	32	86	**196**	13
8	src/contracts/protocol/libraries/logic/ValidationLogic.sol	642	65	133	**444**	16
9	src/contracts/protocol/libraries/types/DataTypes.sol	330	27	68	**235**	0
10	src/contracts/protocol/pool/Pool.sol	901	84	112	**705**	3
	Total	**4212**	**412**	**895**	**2905**	**8**

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

Aave v3.3.0, building on v3.2, introduces improvements targeting better debt management, liquidation logic, efficiency optimizations, and improved integration tools.

Key Features

1. **Bad Debt Management**:
   - Prevents the accumulation of bad debt (debt left unpaid after liquidation) by halting interest accrual on accounts with zero collateral and remaining debt.
   - Remaining bad debt is "burned," and the resulting deficit is accounted for in the reserve. This ensures better protocol sustainability and prevents stale balances.
   - Introduces deficit reduction mechanisms, enabling authorized parties to reduce reserve deficits by burning corresponding aTokens.
2. **Liquidation Logic Updates**:
   - **50% Close Factor Changes**: Now applies to entire positions instead of individual reserves, facilitating more efficient liquidations.
   - **Position-Size-Based Full Liquidations**: Positions smaller than a defined threshold can now be fully liquidated (100% close factor), minimizing the creation of "dust debt" (small leftover debt).
   - **Forced Position Cleanup**: Liquidations now ensure no residual dust debt or leftover collateral below certain thresholds, improving protocol cleanup mechanisms.
3. **Optimization of Bitmap Access**:
   - Reserve bitmasks are optimized for frequent **reads** instead of writes, reducing gas costs during common protocol interactions, especially with multiple positions.
4. **Additional Getters**:
   - New getter functions (e.g., `getReserveAToken()`) reduce gas costs for external integrations by eliminating unnecessary fetches of storage slots.

Breaking Changes

- The deprecated `getReserveDataExtended()` function has been removed.
- Usage of "unbacked tokens" in `calculateInterestRates` now includes both `reserve.deficit` and `reserve.unbacked`.

Impact: These updates improve protocol efficiency, reduce gas costs, and enhance liquidation mechanisms to better manage risks and reduce long-term liabilities.

2.6 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/contracts/protocol/libraries/logic/LiquidationLogic.sol	**3**	0	0	0	3
src/contracts/protocol/libraries/configuration/EModeConfiguration.sol	**1**	0	0	0	1
src/contracts/protocol/libraries/logic/EModeLogic.sol	**1**	0	0	0	1
src/contracts/protocol/libraries/logic/GenericLogic.sol	**1**	0	0	0	1
src/contracts/protocol/libraries/logic/ValidationLogic.sol	**1**	0	0	0	1
src/contracts/protocol/pool/L2Pool.sol	**1**	0	0	0	1
src/contracts/protocol/pool/PoolConfigurator.sol	**1**	0	0	0	1

2.7 CONCLUSION

A comprehensive audit was conducted on 10 smart contracts, initially revealing only 3 informational notes. The audit highlighted a mismatch in how `msg.sender` was passed for interest repayment in handleRepayment, unused imports scattered across multiple contracts, and a minor typo in `LiquidationLogic` comments.

Following our initial audit, Aave worked with our team to address the identified issues. All of them have now been successfully resolved.

As a result, the project has passed our audit. Our auditors have verified that the Aave v3.3.0, as of audited commit `21c30148d1484ddec57f5d223f530179b103cae6`, operates as intended within the defined scope, based on the information and code provided at the time of evaluation.

3 FINDINGS REPORT

OXORIO

3.1 CRITICAL

No critical issues found.

3.2 MAJOR

No major issues found.

3.3 WARNING

No warning issues found.

I-01	`msg.sender` as a parameter in the `handleRepayment` function when repayment is made by the protocol in `LiquidationLogic`
Severity	**INFO**
Status	• FIXED

Location

File	Location	Line
LiquidationLogic.sol	contract `LiquidationLogic` > function `_burnDebtTokens`	577

Description

In the function `_burnDebtTokens` of contract `LiquidationLogic`, the comments indicate that the protocol absorbs interest losses in the event of bad debt in GHO tokens:

```
uint256 amountToBurn = accruedInterest - actualDebtToLiquidate;
// In the case of GHO, all obligations are to the protocol
// therefore the protocol assumes the losses on interest and only tracks the pure deficit by
discounting the not-collected & burned debt
outstandingDebt -= amountToBurn;
IAToken(debtReserveCache.aTokenAddress).handleRepayment(msg.sender, user, amountToBurn);
```

However, the loss write-off for `amountToBurn` is performed by calling the `handleRepayment` function with the same `msg.sender` parameter used for the liquidation of `actualDebtToLiquidate`.

As a result, within the `handleRepayment` function, the liquidator is identified as the user making the interest payment, even though the protocol is actually accounting for the loss itself.

Recommendation

We recommend considering passing the address of the GHO issuer or another relevant address as the first parameter of the `handleRepayment` function to account for accrued interest as repayment for losses, instead of using the liquidator's address.

Update

Fixed in commit `30b78ab15602adfbe3d95569dfee81a88efe090c`.

I-02	Unused imports
Severity	**INFO**
Status	• FIXED

Location

File	Location	Line
EModeConfiguration.sol	-	5
EModeLogic.sol	-	6
GenericLogic.sol	-	4
GenericLogic.sol	-	14
LiquidationLogic.sol	-	13
ValidationLogic.sol	-	7
ValidationLogic.sol	-	8
L2Pool.sol	-	5
PoolConfigurator.sol	-	18
PoolConfigurator.sol	-	19

Description

In the mentioned locations, contracts or libraries are imported but not used in the code.

Recommendation

We recommend removing unused imports from the code to maintain codebase cleanliness.

Update

Fixed in commit `30b78ab15602adfbe3d95569dfee81a88efe090c`.

I-03	Typo in comment in `LiquidationLogic`
Severity	**INFO**
Status	• FIXED

Location

File	Location	Line
LiquidationLogic.sol	contract `LiquidationLogic` > function `executeLiquidationCall`	371

Description

In the function `executeLiquidationCall` of contract `LiquidationLogic`, there is a typo in the comment:

```
// This is by design as otherwise debt debt ceiling would render ineffective if a collateral asset faces bad debt events.
```

Recommendation

We recommend fixing the typo in the comment.

Update

Fixed in commit `30b78ab15602adfbe3d95569dfee81a88efe090c`.

4. APPENDIX

OXORIO

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 FINDINGS CLASSIFICATION REFERENCE

4.2.1 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.

4.2.2 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.

4.3 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