AAVE V3.1 SECURITY AUDIT REPORT

Aug 07, 2024

MixBytes()

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of the Client. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 Security Assessment Methodology

A group of auditors are involved in the work on the audit. The security engineers check the provided source code independently of each other in accordance with the methodology described below:

1. Project architecture review:

- Project documentation review.
- General code review.
- Reverse research and study of the project architecture on the source code alone.

Stage goals
- Build an independent view of the project's architecture.
- Identifying logical flaws.

2. Checking the code in accordance with the vulnerabilities checklist:

- Manual code check for vulnerabilities listed on the Contractor's internal checklist. The Contractor's checklist is constantly updated based on the analysis of hacks, research, and audit of the clients' codes.
- Code check with the use of static analyzers (i.e Slither, Mythril, etc).

Stage goal

Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flash loan attacks etc.).

3. Checking the code for compliance with the desired security model:

- Detailed study of the project documentation.
- Examination of contracts tests.
- Examination of comments in code.
- Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit.
- Exploits PoC development with the use of such programs as Brownie and Hardhat.

Stage goal

Detect inconsistencies with the desired model.

4. Consolidation of the auditors' interim reports into one:

- Cross check: each auditor reviews the reports of the others.
- Discussion of the issues found by the auditors.
- Issuance of an interim audit report.

Stage goals

- Double-check all the found issues to make sure they are relevant and the determined threat level is correct.
- Provide the Client with an interim report.

5. Bug fixing & re-audit:

- The Client either fixes the issues or provides comments on the issues found by the auditors. Feedback from the Customer must be received on every issue/bug so that the Contractor can assign them a status (either "fixed" or "acknowledged").
- Upon completion of the bug fixing, the auditors double-check each fix and assign it a specific status, providing a proof link to the fix.
- A re-audited report is issued.

Stage goals

- Verify the fixed code version with all the recommendations and its statuses.
- Provide the Client with a re-audited report.

6. Final code verification and issuance of a public audit report:

- The Customer deploys the re-audited source code on the mainnet.
- The Contractor verifies the deployed code with the re-audited version and checks them for compliance.
- If the versions of the code match, the Contractor issues a public audit report.

Stage goals

- Conduct the final check of the code deployed on the mainnet.
- Provide the Customer with a public audit report.

Finding Severity breakdown

All vulnerabilities discovered during the audit are classified based on their potential severity and have the following classification:

Severity	Description
Critical	Bugs leading to assets theft, fund access locking, or any other loss of funds.
High	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss funds.
Low	Bugs that do not have a significant immediate impact and could be easily fixed.

Based on the feedback received from the Customer regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future.

1.3 Project Overview

New minor-major 3.1.0 version of Aave v3, focused on different security and operational improvements.

1.4 Project Dashboard

Project Summary

Title	Description
Client	Aave
Project name	Aave v3.1.0
Timeline	April 9 2024 - June 13 2024
Number of Auditors	3

Project Log

Date	Commit Hash	Note
09.04.2024	`ec60c001358ee6ddf316f79461d21f0a1c3ed7a5`	Commit for the audit
27.04.2024	`1af1fc9cc8d4d0b1a7a68a4275c71ab496c3ac34`	Commit for the reaudit
31.05.2024	`17915df8a1ec01911b212fbfcb76cdc00cd7d10a`	Commit with updates
13.06.2024	`d13aef0fba6815480180e30d4f9ee1fb8af5843a`	Commit for the reaudit 2

Project Scope

The audit covered the following files:

in `src/core/contracts/`:

File name	Link

File name	Link
contracts/misc/AaveProtocolDataProvider.sol	AaveProtocolDataProvider.sol
contracts/misc/L2Encoder.sol	L2Encoder.sol
contracts/protocol/libraries/configuration/ReserveConfiguration.sol	ReserveConfiguration.sol
contracts/protocol/libraries/helpers/Errors.sol	Errors.sol
contracts/protocol/libraries/logic/BorrowLogic.sol	BorrowLogic.sol
contracts/protocol/libraries/logic/BridgeLogic.sol	BridgeLogic.sol
contracts/protocol/libraries/logic/ConfiguratorLogic.sol	ConfiguratorLogic.sol
contracts/protocol/libraries/logic/FlashLoanLogic.sol	FlashLoanLogic.sol
contracts/protocol/libraries/logic/LiquidationLogic.sol	LiquidationLogic.sol
contracts/protocol/libraries/logic/PoolLogic.sol	PoolLogic.sol
contracts/protocol/libraries/logic/ReserveLogic.sol	ReserveLogic.sol
contracts/protocol/libraries/logic/SupplyLogic.sol	SupplyLogic.sol
contracts/protocol/libraries/logic/ValidationLogic.sol	ValidationLogic.sol
contracts/protocol/libraries/types/ConfiguratorInputTypes.sol	ConfiguratorInputTypes.sol
contracts/protocol/libraries/types/DataTypes.sol	DataTypes.sol
contracts/protocol/pool/DefaultReserveInterestRateStrategyV2.sol	DefaultReserveInterestRateStrategyV2.sol
contracts/protocol/pool/L2Pool.sol	L2Pool.sol
contracts/protocol/pool/PoolConfigurator.sol	PoolConfigurator.sol
contracts/protocol/pool/Pool.sol	Pool.sol

File name	Link
contracts/protocol/tokenization/AToken.sol	AToken.sol
contracts/protocol/tokenization/StableDebtToken.sol	StableDebtToken.sol
contracts/protocol/tokenization/VariableDebtToken.sol	VariableDebtToken.sol
instances/ATokenInstance.sol	ATokenInstance.sol
instances/L2PoolInstance.sol	L2PoolInstance.sol
instances/PoolConfiguratorInstance.sol	PoolConfiguratorInstance.sol
instances/PoolInstance.sol	PoolInstance.sol
instances/StableDebtTokenInstance.sol	StableDebtTokenInstance.sol
instances/VariableDebtTokenInstance.sol	VariableDebtTokenInstance.sol

Deployments

Ethereum:mainnet

File name	Contract deployed on mainnet	Comment
PoolConfiguratorInstance.sol	0x419226...22b018e5	New implementation for 0x64b761...c39ebb27
PoolInstanceWithCustomInitialize.sol	0x34339f...68c44081	New implementation for 0x87870b...0b4fa4e2
AaveProtocolDataProvider.sol	0x20e074...06113A22	New contract instance

Polygon:mainnet

File name	Contract deployed on mainnet	Comment

File name	Contract deployed on mainnet	Comment
PoolConfiguratorInstance.sol	0x419226...22b018e5	New implementation for 0x8145ed...4510021e
PoolInstanceWithCustomInitialize.sol	0xc4F7b5...d6506477	New implementation for 0x794a61...5b4814ad
AaveProtocolDataProvider.sol	0x7deEB8...4d006eE5	New contract instance

Avalanche:mainnet

File name	Contract deployed on mainnet	Comment
PoolConfiguratorInstance.sol	0x419226...22b018e5	New implementation for 0x8145ed...4510021E
PoolInstanceWithCustomInitialize.sol	0xc4f7b5...d6506477	New implementation for 0x794a61...5b4814aD
AaveProtocolDataProvider.sol	0x7deeb8...4d006ee5	New contract instance

Optimism:mainnet

File name	Contract deployed on mainnet	Comment
PoolConfiguratorInstance.sol	0x419226...22b018e5	New implementation for 0x8145ed...4510021e
PoolInstanceWithCustomInitialize.sol	0x6C6c68...34a3ce47	New implementation for 0x794a61...5b4814ad
AaveProtocolDataProvider.sol	0x7deEB8...4d006eE5	New contract instance

Arbitrum:mainnet

File name	Contract deployed on mainnet	Comment

File name	Contract deployed on mainnet	Comment
PoolConfiguratorInstance.sol	0x419226...22b018e5	New implementation for 0x8145ed...4510021e
PoolInstanceWithCustomInitialize.sol	0x6C6c68...34a3ce47	New implementation for 0x794a61...5b4814ad
AaveProtocolDataProvider.sol	0x7deEB8...4d006eE5	New contract instance

Metis:mainnet

File name	Contract deployed on mainnet	Comment
PoolConfiguratorInstance.sol	0x419226...22b018e5	New implementation for 0x69FEE8...528645A6
PoolInstanceWithCustomInitialize.sol	0x3e89ce...720276ad	New implementation for 0x90df02...Bf1D6a57
AaveProtocolDataProvider.sol	0xd554b5...c46f9fa9	New contract instance

Base:mainnet

File name	Contract deployed on mainnet	Comment
PoolConfiguratorInstance.sol	0x419226...22b018e5	New implementation for 0x5731a0...6e9A96be
PoolInstanceWithCustomInitialize.sol	0x527F60...50113B9a	New implementation for 0xa238dd...3f98d1c5
AaveProtocolDataProvider.sol	0x793177...32184BBC	New contract instance

Gnosis:mainnet

File name	Contract deployed on mainnet	Comment

File name	Contract deployed on mainnet	Comment
PoolConfiguratorInstance.sol	0x419226...22b018e5	New implementation for 0x730497...e5A75D16
PoolInstanceWithCustomInitialize.sol	0xe07E26...49328Dc7	New implementation for 0xb50201...56E326D8
AaveProtocolDataProvider.sol	0xF4EbEE...f87F6C4A	New contract instance

Scroll:mainnet

File name	Contract deployed on mainnet	Comment
PoolConfiguratorInstance.sol	0x419226...22b018e5	New implementation for 0x32BCab...e0Df9b7f
PoolInstanceWithCustomInitialize.sol	0x8281ae...B1D67FE0	New implementation for 0x11fCfe...37D3cFfe
AaveProtocolDataProvider.sol	0xD9b61A...9D688a63	New contract instance

BSC:mainnet

File name	Contract deployed on mainnet	Comment
PoolConfiguratorInstance.sol	0x419226...22b018e5	New implementation for 0x67bdF2...5D6f9584
PoolInstanceWithCustomInitialize.sol	0x30AFbf...57Bb5Eb7	New implementation for 0x6807dc...79a7e0cB
AaveProtocolDataProvider.sol	0x673639...6e5d8d4B	New contract instance

1.5 Summary of findings

Severity	# of Findings
Critical	0
High	0
Medium	0
Low	3

ID	Name	Severity	Status
L-1	Riskless leverage during the grace period	Low	Acknowledged
L-2	No sanity checks on the grace period setup	Low	Fixed
L-3	Frontrun griefing of isolated mode users	Low	Acknowledged

1.6 Conclusion

In this audit, we systematically evaluated the differential update of Aave 3.1, focusing on several critical updates and potential issues, organized into key sections:

1. **User Identification Update**: The audit confirmed the successful replacement of `msg.sender` references with user-specific variables. A single oversight was identified in unreachable code, which does not affect the platform's security or functionality.

2. **Virtual Balances**: Examination of the newly introduced virtual balances showed they are correctly implemented, with checks confirming there are no risks of overflow or underflow, ensuring their safe operation within the ecosystem.

3. **Stable Rate Deprecation**: Addressing a critical vulnerability found last year, the audit noted the deprecation of stable rate calculations. New users are prevented from opting for stable rates, and existing users can be switched to variable rates through a public, permissionless function.

Further, the audit included comprehensive checks to ensure the system's overall integrity and compatibility:

- **Backward Compatibility and Safety Evaluations**: Tests confirmed that V3 implementations are backward compatible with current contracts. Additional checks covered boundary conditions, storage validity, potential side effects of the `AToken.balanceOf` method, permit methods in V3, and the upgrade process for V3 contracts.

- **Security Checklist**: A thorough review against a detailed checklist confirmed no dangerous changes were introduced. The checklist covered aspects such as business logic, common ERC20 issues, share calculation problems, interactions with external contracts, integer overflow, reentrancy attacks, and access control.

We conclude that the provided updates to Aave 3.1 do not introduce new vulnerabilities and fix several old problems.

2.FINDINGS REPORT

2.1 Critical

Not Found

2.2 High

Not Found

2.3 Medium

Not Found

2.4 Low

L-1	Riskless leverage during the grace period
Severity	Low
Status	Acknowledged

Description

The grace period was designed to give users some time to increase their health.

- ValidationLogic.sol#L534-L538

But also, it can be abused as protection from liquidations - the whole grace period can be used as riskless time for more leverage, without the risk of being liquidated.

Recommendation

We recommend disallowing actions that decrease health (new borrowings and withdrawals) during the asset grace period.

Client's commentary

> Setting a liquidations grace period is an action to always be recommended by risk providers, and the period should be as short as possible, prioritasing always the overall health of the system. No extra protection is added on this given that in our opinion the complexity introduced is not worth it.

L-2	No sanity checks on the grace period setup
Severity	Low
Status	Fixed in 1af1fc9c

Description

The grace period can be set to any value without validation:

- PoolLogic.sol#L137

A longer grace period setup will leave markets with a worse debt risk. So, the hardcoded limits will allow managing that risk.

Recommendation

We recommend having an upper limit, e.g., 1 week from the current timestamp.

L-3	Frontrun griefing of isolated mode users
Severity	Low
Status	Acknowledged

Description

There is the following attack vector:

- STEP 1) UserA supplies an asset that is Isolated - CRV. CRV will not automatically be used as collateral, so UserA will have to additionally call `Pool.setUserUseReserveAsCollateral()` later (STEP 3).

- STEP 2) FrontrunnerB supplies a small amount on behalf of userA. The asset is not among Isolated assets - e.g., WETH. This transaction is confirmed before supplying CRV by userA. So, WETH will automatically be used as collateral. This check will return `true` for supplying WETH: ValidationLogic.sol#L726-L741

- STEP 3) UserA calls `Pool.setUserUseReserveAsCollateral()` for CRV.
  But the call will fail and CRV will not be used as collateral as `validateUseAsCollateral()`: SupplyLogic.sol#L271-L277

As a result, UserA will not be able to use CRV as collateral. UserA will have to manually either get rid of aWETH or disable WETH as collateral. It is easy for EOAs, and less easy for smart-contracts that may not expect such problems.

Recommendation

Although the attacker is not directly motivated to do this, we recommend changing the collateral configuration logic to protect against this attack vector.

Client's commentary

> This is not really something fixable in a straightforward manner, or unique to 3.1, because it boils down to the behaviour on transferability of aTokens: enabling as collateral when receiving non-isolated ones.
> The only solution would be removing this behaviour, but we are fairly sure there are applications expected the automatic enabling as collateral, and we should not break them.
> No action.

3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build opensource solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

Contacts


https://github.com/mixbytes/audits_public


https://mixbytes.io/


hello@mixbytes.io


https://twitter.com/mixbytes