MOTECH AUDIT

SMART CONTRACT SECURITY AUDIT

SECURITY ASSESSMENT

2021

AAVE PROGRESS REPORT

31 OCT 2021

SECURITY ASSESMENT

TABLE OF CONTENTS

Finding

SUMMARY

This report has been prepared for Aave to discover issues and vulnerabilities in the source code of theAave project as well as any contract dependencies that were not part of an officially recognized library. Acomprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts producedby industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommendaddressing these findings to ensure a high level of security standards and industry practices. We suggestrecommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified inpublic;
- Provide more transparency on privileged activities once the protocol is live.

BACKGROUND

MotechAudit was commissioned by AAVE Token to perform an audit of smart contracts:
https://etherscan.io/address/0x7fc66500c84a76ad7e9c93437bfc5ac33e2ddae9
The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

SECURITY ASSESMENT

AUDIT DETAILS

AUDITED PROJECT

AAVE

DEPLOYER ADDRESS

0x51F22ac850D29C879367A77D241734AcB276B815

CLIENT CONTACTS:

AAVE Token team

BLOCKCHAIN

ETHEREUM Project

WEBSITE:

https://aave.com/

MOTECH AUDIT

SMART CONTRACT SECURITY AUDIT

DISCLAIMER

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and MotechAudit and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (MotechAudit) owe no duty of care towards you or any other person, nor does MotechAudit make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and MotechAudit hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, MotechAudit hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against MotechAudit, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

SECURITY ASSESMENT

CONTRACT DETAILS

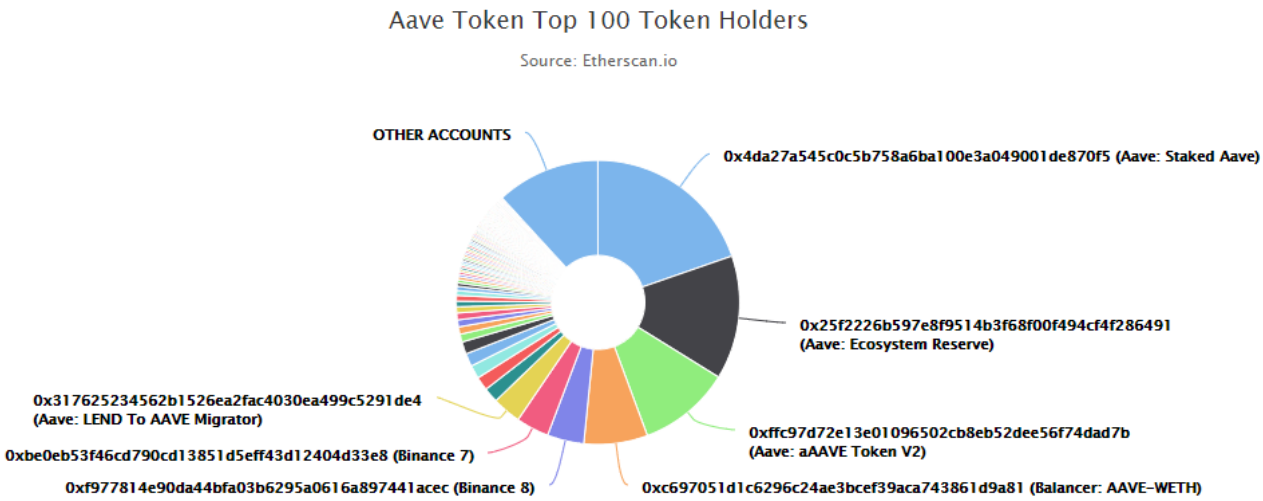
Token contract details for Jul-30-2021

Contract name	**AAVE**
Contract address	**0x7Fc66500c84A76Ad7e9c93437bFc5Ac33E2DDaE9**
Total supply	**16,000,000**
Token ticker	**AAVE Token(AAVE)**
Decimals	**18**
Token holders	**97977**
Transactions count	**1,188,056**
Top 100 holders dominance	**88.1583%**
Contract deployer address	**0x51F22ac850D29C879367A77D241734AcB276B815**
Contract's current owner address	**0x51F22ac850D29C879367A77D241734AcB276B815**

MOTECH AUDIT
SMART CONTRACT SECURITY AUDIT

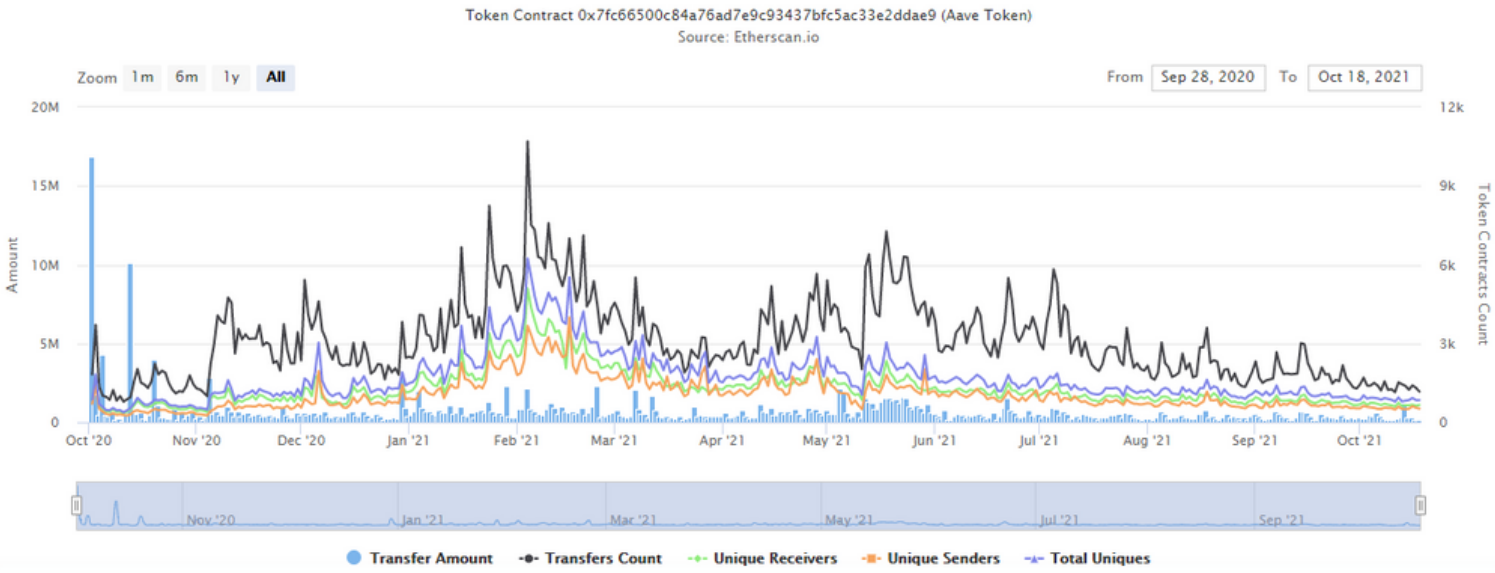
SECURITY ASSESMENT

AAVE TOKEN DISTRIBUTION



(A total of 14,107,742.34 tokens held by the top 100 accounts from the total supply of 16,000,000.00 token)

AAVE TOKEN CONTRACT INTERACTION DETAILS



MOTECH AUDIT
SMART CONTRACT SECURITY AUDIT

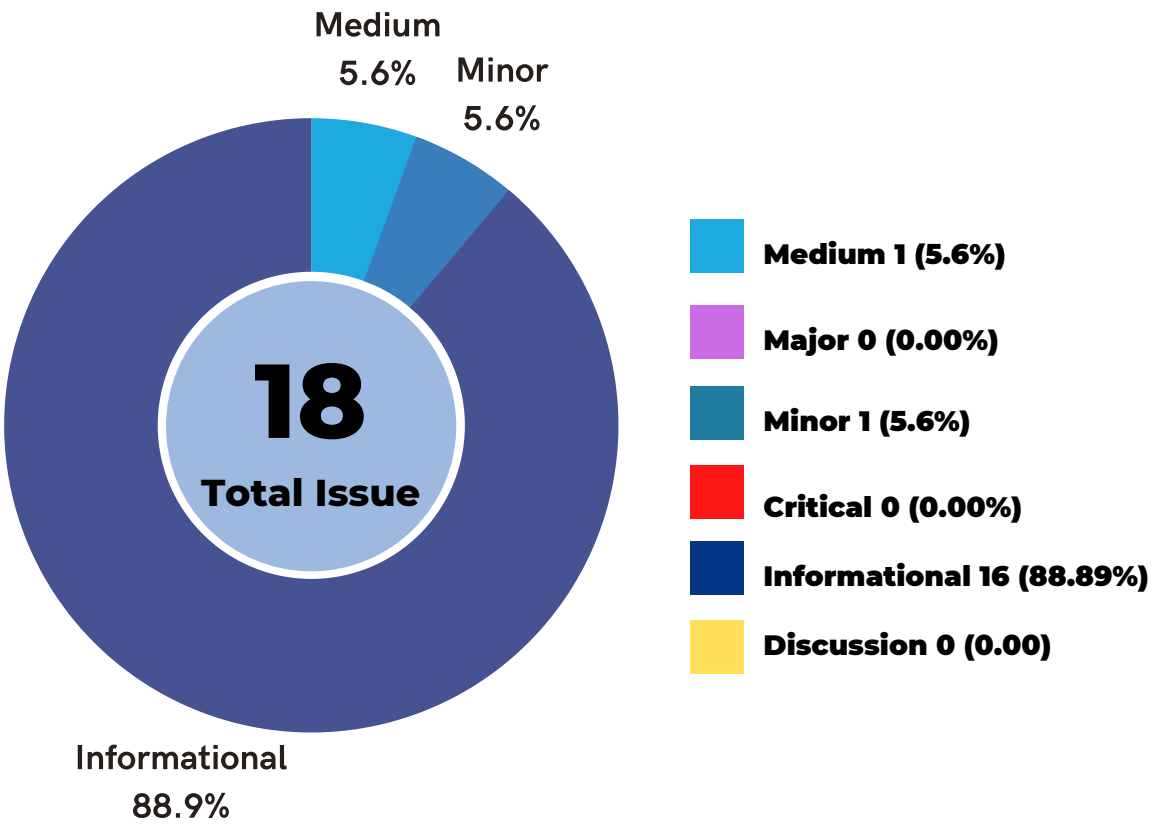
TOP 10 TOKEN HOLDERS

Rank	Address	Quantity	Percentage	Value	Analytics
1	Aave: Staked Aave	3,153,064.149331849582085023	19.7067%	$936,854,252.94	
2	Aave: Ecosystem Reserve	2,257,782.03113548673275894	14.1111%	$670,843,534.38	
3	Aave: aAAVE Token V2	1,682,219.179713368607878452	10.5139%	$499,829,409.82	
4	Balancer: AAVE-WETH	1,165,634.642094147792067741	7.2852%	$346,339,218.01	
5	Binance 8	670,539.15249136	4.1909%	$199,233,960.05	
6	Binance 7	600,000	3.7500%	$178,275,012.86	
7	Aave: LEND To AAVE Migrator	524,270.921484809083351552	3.2767%	$155,774,008.78	
8	Polygon (Matic): ERC20 Bridge	273,044.632261641071144552	1.7065%	$81,128,392.21	
9	SushiSwap: AAVE	249,949.411608303969222636	1.5622%	$74,266,224.28	
10	Avalanche: Bridge	243,452.86727862841656099	1.5216%	$72,335,938.41	

source:etherscan.io

SECURITY ASSESMENT

FINDINGS



ID	Title	Category	Severity	Status
ATC-01	Unlocked Compiler Version	Language Specific	Informational	Resolved
ATC-02	Function & Variable Visibility	Coding Style	Informational	Resolved
ATC-03	Inefficient Greater-Than Comparison w/ Zero	Gas Optimization	Informational	Partially Resolved
ATC-04	Unconventional Naming of `public` Variables	Coding Style	Informational	Acknowledged
ATC-05	EIP712 Adjustment	Coding Style	Informational	Acknowledged
ATC-06	Redundant Assignments & `_setupDecimals` Invocation	Gas Optimization	Informational	Resolved

ID	Title	Category	Severity	Status
ATC-07	Inconsistent EIP2612 Implementation	Logical Issue	Medium	Resolved
ATC-08	Redundant `SafeMath` Utilization	Gas Optimization	Informational	Acknowledged
ATC-09	Declaration Optimization	Gas Optimization	Informational	Resolved
DTH-01	Inexistent Access Control	Logical Issue	Informational	Acknowledged
LTA-01	Unlocked Compiler Version	Language Specific	Informational	Resolved
LTA-02	Function & Variable Visibility	Coding Style	Informational	Resolved
LTA-03	Inefficient Greater-Than Comparison w/ Zero	Gas Optimization	Informational	Partially Resolved
LTA-04	Unconventional Naming of `public` Variables	Coding Style	Informational	Acknowledged
LTA-05	Truncation of LEND Migration Amount	Mathematical Operations	Minor	Acknowledged
LTA-06	Redundant `SafeMath` Utilization	Gas Optimization	Informational	Acknowledged
LTA-07	Purpose of `initializer` Modifier	Gas Optimization	Informational	Acknowledged
VIC-01	Function & Variable Visibility	Coding Style	Informational	Resolved

ATC-01 | UNLOCKED COMPILER VERSION

Category	Severity	Location	Status
Language Specific	Informational	AaveToken.sol: 1	Resolved

Description

The smart contract "pragma" statements regarding the compiler version indicate that version 0.6.10 orhigher should be utilized.

Recommendation

We advise that the compiler version is locked at version 0.6.10 or whichever Solidity version higher thanthat satisfi es the requirements of the codebase as an unlocked compiler version can lead to discrepanciesbetween compilations of the same source code due to compiler bugs and differences.

Alleviation

As per our recommendation, the AAVE team locked both contracts at version 0.6.10 aiding in pinpointingcompiler bugs should they occur.

ATC-02 | FUNCTION & VARIABLE VISIBILITY

Category	Severity	Location	Status
Coding Style	Informational	AaveToken.sol: 31, 121~123	Resolved

Description

The Aave team has applied an adjusted version of the Initializable trait defined in the OpenZeppelinlibraries whereby a revision number is utilized for discerning between initialize deployments.To achieve this, a getRevision function is meant to be implemented as an internal function withinderivative contracts of VersionedInitializable. For this purpose, Aave has defined these functions aswell as declared a REVISION constant that is publicly accessible.

Recommendation

We advise that the function signature of getRevision is instead converted to public. As constantvariables are meant to conform to the UPPER_CASE_FORMAT, a getter function in the form ofgetRevision is more legible and sensible than invoking “REVISION” from off-chain applications.

Alleviation

The AAVE team responded to this Exhibit by stating that the internal styling guideline they conform toutilizes auto-generated getters instead of user-defined ones as they are less error prone and less verboseand as such, this Exhibit is inapplicable.

ATC-03 | INEFFICIENT GREATER-THAN COMPARISON W/ ZERO

Category	Severity	Location	Status
Gas Optimization	Informational	AaveToken.sol: 135	Partially Resolved

Description

The lines above conduct a greater-than > comparison between unsigned integers and the value literal 0

Recommendation

As unsigned integers are restricted to the positive range, it is possible to convert this check to an inequality!= reducing the gas cost of the functions. Additionally, L62 of migrateFromLEND in LendToAaveMigratorcould instead internally call the function migrationStarted which would have to be converted to public.This would ensure consistency in the checks and enable additional checks to be imposed onmigrationStarted in the future if necessary. If no subsequent development is expected, the last point canbe safely ignored as it would lead to a miniscule increase in the gas cost of migrateFromLEND.

Alleviation

The AAVE team evaluated this Exhibit and proceeded with applying the greater-than to inequalityoptimization on the aforementioned lines and chose to avoid declaring migrationStarted as public andretaining L62 as is.

ATC-04 | UNCONVENTIONAL NAMING OF PUBLIC VARIABLES

Category	Severity	Location	Status
Coding Style	Informational	AaveToken.sol: 34, 36, 38, 43, 45, 47	Acknowledged

Description

The variables of L34, L36, L38 and L43 are public yet prefixed with an underscore. TheDOMAIN_SEPARATOR defined in L45 is listed as public yet follows the naming convention of constant andimmutable variables and the PERMIT_TYPEHASH is declared as public correctly following theUPPER_CASE_FORMAT but being illegible for off-chain applications via its compiler-generated getter.

Recommendation

We advise that the first four variable declarations omit the underscore, the DOMAIN_SEPARATOR variable isconverted to the camelCase format and that a dedicated getter is defined for the PERMIT_TYPEHASHvariable which should be set to either internal or private.If Exhibit 7 is followed, the point about DOMAIN_SEPARATOR should be ignored. Additionally, the variable ofL38 could be renamed to snapshotsLength instead of countsSnapshots to aid in understanding itspurpose.

Alleviation

As per the AAVE's response to Exhibit 2 and Exhibit 4, they chose to retain the naming convention as it iscompliant with the team's internal styling guidelines.

ATC-05 | EIP712 ADJUSTMENT

Category	Severity	Location	Status
Coding Style	Informational	AaveToken.sol: 45, 65~71	Acknowledged

Description

The DOMAIN_SEPARATOR variable of EIP712 can be converted to immutable and set within the constructorby utilizing the EIP1344 which was partially created for EIP712.

Recommendation

The assignment of L65 - L71 should instead be moved to the constructor of the contract. As at itscurrent state it relies on the chainId parameter, this can be derived using EIP1344 via assembly by usingthe chainid() opcode. A simple example of utilization would be to declare a uint256 variable titledchainId, declare an assembly block and assign the result of chainid() via the := operator to the variablechainId and subsequently use it as per the original assignment. Additionally, the statement of L68 shouldinstead use the current REVISION of the contract rather than the string literal 1 to ensure that updates inthe codebase are reflected in the EIP712 domain separator.

Alleviation

The AAVE team partially acknowledged this Exhibit by following our recommendation regarding retrievingthe chainId variable of EIP1344 via assembly. Our recommendation with regards to setting theDOMAIN_SEPARATOR as immutable was avoided as the assignment of the variable relies on the statementaddress(this) which would differ when executed in the constructor, resulting in the address of the logiccontract, and when executed in the initialize function, resulting in the address of the proxy contract. Itis still feasible to set the variable as immutable by passing in the address of the proxy to the constructor ofthe AaveToken logic contract, however we leave this optimization up to the discretion of the AAVE team asthey may wish to avoid linking the logic contract with the proxy contract directly. The rationale behind thisoptimization is that gas cost will be greatly reduced for permit invocations as they would not require toread from state and would instead read the resulting literal on the code itself as that is what the immutabletrait does. Additionally, our point with regards to utilizing the REVISION variable instead of the string literal1 on L75 still stands.Similarly to Exhibit 5, the time constraints that the AAVE team had to comply with did not permit majorchanges in the codebase meaning that the team decided not to alter the deployment procedure of thecontracts after weighing the benefit. Additionally, with regards to the REVISION variable, the team stated that they decided to keep (the original implementation) to avoid extra encoding / cast fromthe uint256 REVISION.

ATC-06 | REDUNDANT ASSIGNMENTS & _SETUPDECIMALS INVOCATION

Category	Severity	Location	Status
Gas Optimization	Informational	AaveToken.sol: 72~74	Resolved

Description

The function _setupDecimals is called to set the decimals of the ERC20 interface to 18. Internally, theERC20 interface assigns the literal 18 by default to the value of decimals. Additionally, the values of NAMEand SYMBOL are set to _name and _symbol respectively whereas the constructor of ERC20 is called on L51which assigns those values as well.

Recommendation

All aforementioned statements can be safely omitted as they are duplicate assignments.

Alleviation

The AAVE team articulated how both contracts are meant to be utilized via a proxy and as such, the pointswith regards to the redundant assignments are void. The team addressed the _setupDecimals invocationby utilizing a constant DECIMALS variable instead of the value literal 18.

ATC-07 | INCONSISTENT EIP2612 IMPLEMENTATION

Category	Severity	Location	Status
Logical Issue	Medium	AaveToken.sol: 47, 91~116	Resolved

Description

The typehash of the permit function as well as its internal statements do not conform to the EIP2612specification as the expiration and value variables are swapped in between implementations.

Recommendation

We advise that EIP2612 is conformed to the letter, as the current implementation is misleading. The reasonit is misleading is because the function signature matches the specification's ABI (address, address,uint256, uint256, uint8, bytes32, bytes32) yet the typehash utilized is different as well as the waythe values are utilized. This could lead to a generic EIP2612 implementation misleading users to input thevalue to be transacted and the deadline whilst those two will be utilized in place of one another withinthe codebase. As EIPs are meant to streamline the way smart contracts interface with off chainapplications on the Solidity ecosystem, we advise that the lines of this Exhibit are amended accordingly toconform to EIP2612.

Alleviation

The AAVE team fully addressed this Exhibit in a dedicated merge-request as it necessitated changesthroughout the codebase as well as the test suites.

ATC-08 | REDUNDANT SAFEMATH UTILIZATION

Category	Severity	Location	Status
Gas Optimization	Informational	AaveToken.sol: 114, 135, 136, 139, 157, 161	Acknowledged

Description

The SafeMath library has been defined to enforce bound checking for any additions and / or subtractionsthat occur using its exposed methods such as add and sub. All lines mentioned by this Exhibit either haveguaranteed bounds by surrounding code or are logically safe.

Recommendation

The safe addition conducted on L114 for the currentValidNonce of an address will never reach the limit ofuint256 as it is sequentially increasing by one and the number of permit transactions necessary to reachit are practically infinite. The lines 135, 136 and 139 of the _writeSnapshot function refer to the number ofsnapshots stored in the mapping(uint256 => Snapshot) variable. As this by itself has an inherent limit ofuint256, safe additions are unnecessary as the number of Snapshots is practically infinite. Additionally,the safe subtractions that occur on L135 and L136 are done so by subtracting the literal 1 and arepreceded by a bound checking ensuring the variable being subtracted from, ownerCountOfSnapshots, isabove zeroThe safe subtractions and additions done on L157 and L161 of _beforeTokenTransfer are able to indeedunderflow or overflow, however this type of underflow and / or overflow is captured by the actual transferoperations occurring with ERC20 itself so they can be safely omitted as the ERC20 transfer will revertcausing all previous operations to be reverted as well. Finally, the safe division done in L66 ofLendToAaveMigrator is redundant as it simply checks that the right hand of the division is non-zero, a traitguaranteed by what LEND_AAVE_RATIO is meant to represent. This makes the import of SafeMath entirelyredundant in LendToAaveMigrator.

Alleviation

The AAVE team acknowledged the statements of this Exhibit, however they chose to not apply theoptimizations as they desire to keep consistency in the way calculations are carried out throughout theirproject as well as reduce any potential reviewer's cognitive effort in evaluating the code the statements aresurrounded by.

ATC-09 | DECLARATION OPTIMIZATION

Category	Severity	Location	Status
Gas Optimization	Informational	AaveToken.sol: 114, 135, 136, 139, 157, 161	Acknowledged

Description

The SafeMath library has been defined to enforce bound checking for any additions and / or subtractionsthat occur using its exposed methods such as add and sub. All lines mentioned by this Exhibit either haveguaranteed bounds by surrounding code or are logically safe.

Recommendation

The safe addition conducted on L114 for the currentValidNonce of an address will never reach the limit ofuint256 as it is sequentially increasing by one and the number of permit transactions necessary to reachit are practically infinite. The lines 135, 136 and 139 of the _writeSnapshot function refer to the number ofsnapshots stored in the mapping(uint256 => Snapshot) variable. As this by itself has an inherent limit ofuint256, safe additions are unnecessary as the number of Snapshots is practically infinite. Additionally,the safe subtractions that occur on L135 and L136 are done so by subtracting the literal 1 and arepreceded by a bound checking ensuring the variable being subtracted from, ownerCountOfSnapshots, isabove zeroThe safe subtractions and additions done on L157 and L161 of _beforeTokenTransfer are able to indeedunderflow or overflow, however this type of underflow and / or overflow is captured by the actual transferoperations occurring with ERC20 itself so they can be safely omitted as the ERC20 transfer will revertcausing all previous operations to be reverted as well. Finally, the safe division done in L66 ofLendToAaveMigrator is redundant as it simply checks that the right hand of the division is non-zero, a traitguaranteed by what LEND_AAVE_RATIO is meant to represent. This makes the import of SafeMath entirelyredundant in LendToAaveMigrator.

Alleviation

The AAVE team acknowledged the statements of this Exhibit, however they chose to not apply theoptimizations as they desire to keep consistency in the way calculations are carried out throughout theirproject as well as reduce any potential reviewer's cognitive effort in evaluating the code the statements aresurrounded by.

DTH-01 | INEXISTENT ACCESS CONTROL

Category	Severity	Location	Status
Logical Issue	Informational	DoubleTransferHelper.sol: 14~17	Acknowledged

Description

The function doubleSend of the DoubleTransferHelper contract conducts two transfer operations withthe funds owned by the contract.

Recommendation

We advise that proper access control is imposed on this function as at its current state any party is able toinvoke it. The reason this finding was labelled as informational is because DoubleTransferHelper is atest suite dependency and as such we expect it to not be utilized in a production context.

Alleviation

The AAVE team acknowledged this Exhibit and will not apply it as it is solely a test suite dependency forevaluating two snapshots created within the same block.

LTA-01 | UNLOCKED COMPILER VERSION

Category	Severity	Location	Status
Language Specific	Informational	LendToAaveMigrator.sol: 1	Resolved

Description

The smart contract "pragma" statements regarding the compiler version indicate that version 0.6.10 orhigher should be utilized

Recommendation

We advise that the compiler version is locked at version 0.6.10 or whichever Solidity version higher thanthat satisfi es the requirements of the codebase as an unlocked compiler version can lead to discrepanciesbetween compilations of the same source code due to compiler bugs and differences.

Alleviation

As per our recommendation, the AAVE team locked both contracts at version 0.6.10 aiding in pinpointingcompiler bugs should they occur.

LTA-02 | FUNCTION & VARIABLE VISIBILITY

Category	Severity	Location	Status
Coding Style	Informational	LendToAaveMigrator.sol: 20, 74~76	Resolved

Description

The Aave team has applied an adjusted version of the Initializable trait defined in the OpenZeppelinlibraries whereby a revision number is utilized for discerning between initialize deployments.To achieve this, a getRevision function is meant to be implemented as an internal function withinderivative contracts of VersionedInitializable. For this purpose, Aave has defined these functions aswell as declared a REVISION constant that is publicly accessible.

Recommendation

We advise that the function signature of getRevision is instead converted to public. As constantvariables are meant to conform to the UPPER_CASE_FORMAT, a getter function in the form ofgetRevision is more legible and sensible than invoking “REVISION” from off-chain applications.

Alleviation

The AAVE team responded to this Exhibit by stating that the internal styling guideline they conform toutilizes auto-generated getters instead of user-defined ones as they are less error prone and less verboseand as such, this Exhibit is inapplicable.

LTA-03 | INEFFICIENT GREATER-THAN COMPARISON W/ ZERO

Category	Severity	Location	Status
Gas Optimization	Informational	LendToAaveMigrator.sol: 52, 62	Partially Resolved

Description

The lines above conduct a greater-than > comparison between unsigned integers and the value literal 0.

Recommendation

As unsigned integers are restricted to the positive range, it is possible to convert this check to an inequality!= reducing the gas cost of the functions. Additionally, L62 of migrateFromLEND in LendToAaveMigratorcould instead internally call the function migrationStarted which would have to be converted to public.This would ensure consistency in the checks and enable additional checks to be imposed onmigrationStarted in the future if necessary. If no subsequent development is expected, the last point canbe safely ignored as it would lead to a miniscule increase in the gas cost of migrateFromLEND.

Alleviation

The AAVE team evaluated this Exhibit and proceeded with applying the greater-than to inequalityoptimization on the aforementioned lines and chose to avoid declaring migrationStarted as public andretaining L62 as is.

LTA-04 | UNCONVENTIONAL NAMING OF PUBLIC VARIABLES

Category	Severity	Location	Status
Coding Style	Informational	LendToAaveMigrator.sol: 19, 22	Acknowledged

Description

The variable of L19 (LEND_AAVE_RATIO) properly conforms to the naming convention of immutable andconstant variables yet is declared public. The variable of L22 does not conform to the namingconvention of publicly accessible variables as it is prefixed with an underscore.

Recommendation

For the former, we advise that it is instead set to private or internal and a dedicated getter function isset that allows off-chain applications to retrieve the ratio. The Solidity compiler automatically generatesgetter functions for public variables and as such, the above statements are equivalent to the current codein terms of gas impact. For the latter, we advise that the underscore is simply removed from the variabledeclaration.

Alleviation

The AAVE team proceeded with not changing the aforementioned naming conventions of the variables asthe former, L19, is answered by the Alleviation chapter of Exhibit 2, and the latter once again is a resultof AAVE's internal styling guide mandating that the _ prefix is set on all state variables regardless ofvisibility

LTA-05 | TRUNCATION OF LEND MIGRATION AMOUNT

Category	Severity	Location	Status
Mathematical Operations	Minor	LendToAaveMigrator.sol: 64~67	Acknowledged

Description

The migrateFromLEND function accepts an amount input in the form of a uint256, transfers the fullamount of LEND to the contract and subsequently transfers the AAVE equivalent to the sender by dividingthe amount with the LEND_AAVE_RATIO variable.

Recommendation

The division with LEND_AAVE_RATIO will always truncate if LEND_AAVE_RATIO is different than 1 and wouldlead to the truncated amount being permanently locked within LendToAaveMigrator, thus causing the finalconversion ratio to be different than the imposed one. We advise that the modulo of amount withLEND_AAVE_RATIO is subtracted from itself to calculate the exact amount of LEND that will be migrated andprevent trailing LEND from being locked up in the contract.

Alleviation

The documentation material provided to us by AAVE indicated that the team was aware of this truncationmechanism and that the solutions they have investigated are not ideal for this type of issue due to theircomplexity. The important thing here to note is that trailing LEND as well as trailing AAVE will be locked upin the contract as the LEND token is expected to be fully converted to the AAVE token, meaningtruncations will lead to units of both LEND and AAVE remaining out of circulation forever. As this isundesirable behavior, a novel solution we propose would be to instead store the surplus LEND within avariable of the contract that is carried over to the next conversion. This will ensure that when the finaltransfer of LEND to AAVE occurs, no trailing LEND will remain within the contract. Additionally, thiscalculation could also initialize the remainder variable at a value that allows the total supply of LEND to befully divisible by the LEND_AAVE_RATIO variable.The AAVE team took note of our proposed solution, however, due to imminent deadlines theyunderstandably preferred to stick to the original system that has also been properly vetted. We can safelystate that the amount that may be locked up in the contract is indeed minuscule and as such should be ofno concern.

LTA-06 | REDUNDANT SAFEMATH UTILIZATION

Category	Severity	Location	Status
Gas Optimization	Informational	LendToAaveMigrator.sol: 15, 66	Acknowledged

Description

The SafeMath library has been defined to enforce bound checking for any additions and / or subtractionsthat occur using its exposed methods such as add and sub. All lines mentioned by this Exhibit either haveguaranteed bounds by surrounding code or are logically safe.

Recommendation

The safe addition conducted on L114 for the currentValidNonce of an address will never reach the limit ofuint256 as it is sequentially increasing by one and the number of permit transactions necessary to reachit are practically infinite. The lines 135, 136 and 139 of the _writeSnapshot function refer to the number ofsnapshots stored in the mapping(uint256 => Snapshot) variable. As this by itself has an inherent limit ofuint256, safe additions are unnecessary as the number of Snapshots is practically infinite. Additionally,the safe subtractions that occur on L135 and L136 are done so by subtracting the literal 1 and arepreceded by a bound checking ensuring the variable being subtracted from, ownerCountOfSnapshots, isabove zeroThe safe subtractions and additions done on L157 and L161 of _beforeTokenTransfer are able to indeedunderflow or overflow, however this type of underflow and / or overflow is captured by the actual transferoperations occurring with ERC20 itself so they can be safely omitted as the ERC20 transfer will revertcausing all previous operations to be reverted as well. Finally, the safe division done in L66 ofLendToAaveMigrator is redundant as it simply checks that the right hand of the division is non-zero, a traitguaranteed by what LEND_AAVE_RATIO is meant to represent. This makes the import of SafeMath entirelyredundant in LendToAaveMigrator.

Alleviation

The AAVE team acknowledged the statements of this Exhibit, however they chose to not apply theoptimizations as they desire to keep consistency in the way calculations are carried out throughout theirproject as well as reduce any potential reviewer's cognitive effort in evaluating the code the statements aresurrounded by.

LTA-07 | PURPOSE OF INITIALIZER MODIFIER

Category	Severity	Location	Status
Gas Optimization	Informational	LendToAaveMigrator.sol: 45~46	Acknowledged

Description

The initializer modifier as provided by the defacto OpenZeppelin library acts as a guard againstmultiple executions of a function that initializes a contract's values

Recommendation

As the initialize function of LendToAaveMigrator is empty, the initializer modifier could instead beused on the constructor of the contract. This would allow the removal of the initialize function as itserves no purpose at its current state. If it is envisioned that the initialize function will contain code in anext iteration, or that the initialize function is needed to be present in the ABI by your deploymentprocesses, then feel free to ignore this Exhibit and retain the code as is.

Alleviation

As per the second paragraph of our recommendations, the AAVE team responded by stating they expectto add more initializing code in the initialize function for potential next iterations where it would containlogic which is expected.

VIC-01 | FUNCTION & VARIABLE VISIBILITY

Category	Severity	Location	Status
Coding Style	Informational	VersionedInitializable.sol: 39	Resolved

Description

The Aave team has applied an adjusted version of the Initializable trait defined in the OpenZeppelinlibraries whereby a revision number is utilized for discerning between initialize deployments.To achieve this, a getRevision function is meant to be implemented as an internal function withinderivative contracts of VersionedInitializable. For this purpose, Aave has defined these functions aswell as declared a REVISION constant that is publicly accessible

Recommendation

We advise that the function signature of getRevision is instead converted to public. As constantvariables are meant to conform to the UPPER_CASE_FORMAT, a getter function in the form ofgetRevision is more legible and sensible than invoking “REVISION” from off-chain applications.

Alleviation

The AAVE team responded to this Exhibit by stating that the internal styling guideline they conform toutilizes auto-generated getters instead of user-defined ones as they are less error prone and less verboseand as such, this Exhibit is inapplicable.

CONCLUSION

Smart contracts contain owner privileges!

Motech Audit note: Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.