AAVE GOVERNANCE CROSSCHAIN BRIDGES SMART CONTRACT AUDIT

June 17, 2021

MixBytes()

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 Aave. 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 PROJECT OVERVIEW

This scope of contracts contains the crosschain governance bridges used for the aave markets deployed across different networks.

1.3 SECURITY ASSESSMENT METHODOLOGY

At least 2 auditors are involved in the work on the audit who check the provided source code independently of each other in accordance with the methodology described below:

01 "Blind" audit includes:
> Manual code study
> "Reverse" research and study of the architecture of the code based on the source code only

Stage goal:
Building an independent view of the project's architecture
Finding logical flaws

02 Checking the code against the checklist of known vulnerabilities includes:
> Manual code check for vulnerabilities from the company's internal checklist
> The company's checklist is constantly updated based on the analysis of hacks, research and audit of the clients' code

Stage goal:
Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flashloan attacks, etc.)

03 Checking the logic, architecture of the security model for compliance with the desired model, which includes:
> Detailed study of the project documentation
> Examining contracts tests
> Examining comments in code
> Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of the reports from all auditors into one common interim report document
> Cross check: each auditor reviews the reports of the others
> Discussion of the found issues by the auditors
> Formation of a general (merged) report

Stage goal:
Re-check all the problems for relevance and correctness of the threat level
Provide the client with an interim report

05 Bug fixing & re-check.
> Client fixes or comments on every issue
> Upon completion of the bug fixing, the auditors double-check each fix and set the statuses with a link to the fix

Stage goal:
Preparation of the final code version with all the fixes

06 Preparation of the final audit report and delivery to the customer.

Findings discovered during the audit are classified as follows:

FINDINGS SEVERITY BREAKDOWN

Level	Description	Required action
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party	Immediate action to fix issue
Major	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.	Implement fix as soon as possible
Warning	Bugs that can break the intended contract logic or expose it to DoS attacks	Take into consideration and implement fix in certain period
Comment	Other issues and recommendations reported to/acknowledged by the team	Take into consideration

Based on the feedback received from the Customer's team 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 project team is aware of this finding. Recommendations for this finding are planned to be resolved in the future. This finding does not affect the overall safety of the project.
No issue	Finding does not affect the overall safety of the project and does not violate the logic of its work.

1.4 EXECUTIVE SUMMARY

The smart contracts, examined in this audit, are designed to operate on the Polygon and Arbitrum blockchains. The functionality is designed to work with tasks for calling functions in other contracts. You can queue, execute, or cancel tasks. All tasks are saved in a smart contract.

1.5 PROJECT DASHBOARD

Client	Aave
Audit name	Governance Crosschain Bridges
Initial version	7f56e7ae63f30ba8dcd7ced6a11a34c2eb865a1d 763ef5da8befff3a129443a3ff4ef7ca4d3bb446
Final version	763ef5da8befff3a129443a3ff4ef7ca4d3bb446
SLOC	260
Date	2021-06-02 - 2021-06-17
Auditors engaged	2 auditors

FILES LISTING

BridgeExecutorBase.sol	BridgeExecutorBase.sol
ArbitrumBridgeExecutor.sol	ArbitrumBridgeExecuto...
PolygonBridgeExecutor.sol	PolygonBridgeExecutor.sol
IBridgeExecutor.sol	IBridgeExecutor.sol
IFxMessageProcessor.sol	IFxMessageProcessor.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	0
Warning	3
Comment	2

CONCLUSION

Smart contracts have been audited and several suspicious places have been spotted. During the audit no critical or major issues were found, several warnings and comments were spotted. After working on the reported findings all of them were either fixed by the client or acknowledged (if the problem was not critical). So, the contracts are assumed as secure to use according to our security criteria.Final commit identifier with all fixes: `763ef5da8befff3a129443a3ff4ef7ca4d3bb446`

2. FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

Not Found

2.3 WARNING

WRN-1	No validation of the address parameter value in contract constructor
File	BridgeExecutorBase.sol PolygonBridgeExecutor.sol ArbitrumBridgeExecutor.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

The variable is assigned the value of the constructor input parameter. But this parameter is not checked before this. If the value turns out to be zero, then it will be necessary to redeploy the contract, since there is no other functionality to set this variable.

- At the line BridgeExecutorBase.sol#L41 the `_guardian` variable is set to the value of the `guardian` input parameter.
- At the line PolygonBridgeExecutor.sol#L21 the `_fxRootSender` variable is set to the value of the `fxRootSender` input parameter.
- At the line PolygonBridgeExecutor.sol#L22 the `_fxChild` variable is set to the value of the `fxChild` input parameter.
- At the line ArbitrumBridgeExecutor.sol#L18 the `_ethereumGovernanceExecutor` variable is set to the value of the `ethereumGovernanceExecutor` input parameter.

RECOMMENDATION

It is necessary to add a check of the input parameter to zero before initializing the variables.

CLIENT'S COMMENTARY

I think not validating against the 0 address is an acceptable risk. Worst case, you re-deploy. You can't check for all incorrect addresses.

WRN-2	Missing validation on relation
File	BridgeExecutorBase.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At the lines BridgeExecutorBase.sol#L34-L39 are working with the variables `minimumDelay` and `maximumDelay`. But nowhere is there a comparison of these variables with each other.

RECOMMENDATION

It is recommended to add a check for comparing the values of variables between each other.

CLIENT'S COMMENTARY

While we do not directly compare the min and max delay values, we do compare the delay to both the min and the max. If the min and max did not have an appropriate relationship, there would be no delay value that would satisfy both of these lines 34 and 35 in the BaseBridgeExecutor.

WRN-3	The value is assigned to a variable, but not used
File	BridgeExecutorBase.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At the line BridgeExecutorBase.sol#L202 sets the variable `_queuedActions[actionHash]` to `true` when tasks are queued.
At the line BridgeExecutorBase.sol#L269 sets the variable `_queuedActions[actionHash]` to `false` to cancel the job.
But when executed on line BridgeExecutorBase.sol#L235, no validation is made for the `_queuedActions[actionHash]` variable.

RECOMMENDATION

It is recommended to add a check for the value of the `_queuedActions[actionHash]` variable before executing `delegatecall` and `call`.

CLIENT'S COMMENTARY

We perform the action hash in-order to check that the action is not duplicated prior to queuing the action. This occurs in the isActionQueued check of _queue. On execution, if the entire ActionsSet is queued per the check in line 51, then all of it's actions are inherently queued in _queuedActions. therefore checking the _queuedActions mapping for each action prior to executing would never return false.

2.4 COMMENTS

CMT-1	Caching the value will improve the code
File	BridgeExecutorBase.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At the lines BridgeExecutorBase.sol#L176-L183
the calculation of the same value is used many times. But the value of `targets.length` is easier to calculate only once at the very beginning and store it in a variable.
Then work with this variable.

RECOMMENDATION

It is recommended to optimize the code to use the cached value of the variable.

CLIENT'S COMMENTARY

Agree, this would be marginally more optimal, but we are ok with how it is currently implemented. This also mirrors the implementation in Aave-Governance-v2 that is already deployed

CMT-2	Confusing variable name
File	BridgeExecutorBase.sol
Severity	Comment
Status	Fixed at 763ef5da

DESCRIPTION

At the line BridgeExecutorBase.sol#L124, the function is called `getActionsSetState()`. But it is very difficult to understand when in one word there are two different concepts of `get` and `set` at once.
For example, the name `getCurrentState()` will be much clearer.

RECOMMENDATION

It is recommended to rename this variable.

3. ABOUT MIXBYTES

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

BLOCKCHAINS

 Ethereum

 Cosmos

 EOS

 Substrate

TECH STACK

 Python

 Solidity

 Rust

 C++

CONTACTS

 https://github.com/mixbytes/audits_public

 https://mixbytes.io/

 hello@mixbytes.io

 https://t.me/MixBytes

 https://twitter.com/mixbytes