AAVE GOVERNANCE CROSSCHAIN BRIDGES SMART CONTRACT AUDIT

August 12, 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

A group of 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 Project architecture review:
> Reviewing project documentation
> General code review
> Reverse research and study of the architecture of the code based on the source code only
> Mockup prototyping

Stage goal:
Building an independent view of the project's architecture and identifying logical flaws in the code.

02 Checking the code against the checklist of known vulnerabilities:
> 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
> Checking with static analyzers (i.e Slither, Mythril, etc.)

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

03 Checking the code for compliance with the desired security model:
> 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
> Exploits PoC development using Brownie

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of interim auditor reports into a general one:
> 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 and 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 blockchain. 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	9fd0609a2e14d546885f76211961f251d2e15cb9
Final version	0eb2a492e22199bb5746056b3dbf1e861fd7a86b
Date	August 02, 2021 - August 12, 2021
Auditors engaged	2 auditors

FILES LISTING

BridgeExecutorBase.sol	https://github.com/aave/governance-crosschain-bridges/blob/9fd0609a2e14d546885f76211961f251d2e15cb9/contracts/BridgeExecutorBase.sol
PolygonBridgeExecutor.sol	https://github.com/aave/governance-crosschain-bridges/blob/9fd0609a2e14d546885f76211961f251d2e15cb9/contracts/PolygonBridgeExecutor.sol
IBridgeExecutor.sol	https://github.com/aave/governance-crosschain-bridges/blob/9fd0609a2e14d546885f76211961f251d2e15cb9/contracts/interfaces/IBridgeExecutor.sol
IFxMessageProcessor.sol	https://github.com/aave/governance-crosschain-bridges/blob/9fd0609a2e14d546885f76211961f251d2e15cb9/contracts/interfaces/IFxMessageProcessor.sol
SafeMath.sol	https://github.com/aave/governance-crosschain-bridges/blob/9fd0609a2e14d546885f76211961f251d2e15cb9/contracts/dependencies/utilities/SafeMath.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	0
Warning	3
Comment	3

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 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: `0eb2a492e22199bb5746056b3dbf1e861fd7a86b`

2. FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

Not Found

2.3 WARNING

WRN-1	Extra parameter for the method
File	PolygonBridgeExecutor.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At the line PolygonBridgeExecutor.sol#L35, when calling the `processMessageFromRoot()` method, the `stateId` parameter is passed.
But then this variable is not used anywhere in the method.

RECOMMENDATION

It is recommended to remove the unused parameter.

CLIENT'S COMMENTARY

`processMessageFromRoot()` is called in the FxChild contract which we do not control, so we need to receive the parameter, even though we do not do anything with it.

WRN-2	Missing zero address check in constructor
File	BridgeExecutorBase.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At the line BridgeExecutorBase.sol#L44
missing validation `_guardian` for zero address. It is important because contract has no tool to change `_guardian`.

RECOMMENDATION

It is recommended to add zero address validation.

CLIENT'S COMMENTARY

Acknowledge, but no action.

WRN-3	Immutable variable in access modifier
File	BridgeExecutorBase.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At the line BridgeExecutorBase.sol#L22 used the `_guardian` variable in the access modifier. But it is initialized only once in the constructor and there is no other functionality to change. This can lead to unavailability to use `cancel()` method.

RECOMMENDATION

It is recommended to add tool to change `_guardian` variable.

CLIENT'S COMMENTARY

Acknowledge, but no action.

2.4 COMMENT

CMT-1	No validation of the address variable during initialization
File	PolygonBridgeExecutor.sol PolygonBridgeExecutor.sol#70
Severity	Comment
Status	Acknowledged

DESCRIPTION

At the lines
PolygonBridgeExecutor.sol#L61
and
PolygonBridgeExecutor.sol#70
changes variables without validations on same addresses and zero address.

RECOMMENDATION

It is recommended to add validation on same addresses and zero addresses:

PolygonBridgeExecutor.sol#L61

```
require((fxRootSender != _fxRootSender) && (fxRootSender != address(0x0)));
```

PolygonBridgeExecutor.sol#L70

```
require((fxChild != _fxChild) && (fxChild != address(0x0)));
```

CLIENT'S COMMENTARY

Acknowledge, but no action.

CMT-2	Inappropriate documentation for the `receiveFunds()` function
File	BridgeExecutorBase.sol
Severity	Comment
Status	Fixed at 0eb2a492

DESCRIPTION

At the line
BridgeExecutorBase.sol#L122
documentation tag `@inheritdoc` references interface `IBridgeExecutor`, but the interface does not contain a function that is overridden by this function.

RECOMMENDATION

It is recommended to add data to `IBridgeExecutor`.

CMT-3	Bad use of a variable
File	PolygonBridgeExecutor.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At the line PolygonBridgeExecutor.sol#L39, when calling the `processMessageFromRoot()` method, the values of the `rootMessageSender` parameter and the private variable `_fxRootSender` are compared.
It would be logical to use this functionality if the value of the private variable `_fxRootSender` is unknown to anyone.
But due to the existence of the `getFxRootSender()` function, any user can read the value of the `_fxRootSender` variable.
This variable is not used anywhere else in the method. And it is not used in other functions either.
It can be assumed that this variable is only needed for additional validation when calling the `processMessageFromRoot ()` method.
But it is more correct to do this check before calling the `processMessageFromRoot()` method.

RECOMMENDATION

It is recommended to transfer the `fxRootSender` variable from the PolygonBridgeExecutor.sol contract to the FxChild.sol contract.
In this case, the logic of using this variable will be clear.

CLIENT'S COMMENTARY

This functionality is specific to the Polygon bridge and the FxPortal. Because FxChild is part of the FxPortal contracts managed by polygon, the check cannot occur in this contract as it is already deployed.

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