AAVE PARASWAP ADAPTER SMART CONTRACT AUDIT

May 25, 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

Aave is a decentralized non-custodial liquidity markets protocol where users can participate as depositors or borrowers. Depositors provide liquidity to the market to earn a passive income, while borrowers are able to borrow in an overcollateralized (perpetually) or undercollateralized (one-block liquidity) fashion. The audited scope is a part of Aave protocol V2.

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 work with ParaSwap. ParaSwap is a liquidity aggregator from decentralized exchanges. Smart contracts are designed to implement the adapter between AAVE and ParaSwap.

1.5 PROJECT DASHBOARD

Client	Aave
Audit name	ParaSwap Adapter
Initial version	14e2ab47d95f42ec5ee486f367067e78a7588878 4fe36c8fa4c470e2553a4e6632a7d4cc544e5e4c
Final version	4fe36c8fa4c470e2553a4e6632a7d4cc544e5e4c
SLOC	129
Date	2021-05-05 - 2021-05-25
Auditors engaged	2 auditors

FILES LISTING

BaseParaSwapAdapter.sol	BaseParaSwapAdapter.sol
BaseParaSwapSellAdapter.sol	BaseParaSwapSellAdapt...
ParaSwapLiquiditySwapAdapter.sol	ParaSwapLiquiditySwap...

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	0
Warning	5
Comment	5

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: `4fe36c8fa4c470e2553a4e6632a7d4cc544e5e4c`

2.FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

Not Found

2.3 WARNING

WRN-1	You can perform a vulnerable code
File	ParaSwapLiquiditySwapAdapter.sol BaseParaSwapSellAdapter.sol
Severity	Warning
Status	Fixed at 4fe36c8f

DESCRIPTION

At the line

- ParaSwapLiquiditySwapAdapter.sol#L87
  any user can call the `swapAndDeposit ()` function. The parameters `augustus` and `swapCalldata` can be anything. These parameters are used to call the `_sellOnParaSwap ()` function.
  At the line
- BaseParaSwapSellAdapter.sol#L76 in the body of the function `_sellOnParaSwapSwap ()` a smart contract `augustus` is called.
  The user can call any external function for any other smart contract. This smart contract may contain a vulnerable code.

RECOMMENDATION

It is recommended to do the following:

- set the address of the smart contract `augustus` once in the storage variable and control the names of the called methods;
- or interact with `augustus` using the interface.

CLIENT'S COMMENTARY

We created an on chain registry of valid Augustus addresses and calls this from the adapter to validate the input address. Any calldata can be used to call any method of Augustus.

WRN-2	Reentry guard is not used
File	BaseParaSwapAdapter.sol BaseParaSwapSellAdapter.sol
Severity	Warning
Status	Fixed at 9d1cb50d

DESCRIPTION

At the lines:

- BaseParaSwapAdapter.sol#L79
- BaseParaSwapSellAdapter.sol#L35
  it would be robust to ensure that places which assumed to be called once, called once.

RECOMMENDATION

It is recommended to use reentry guard.

CLIENT'S COMMENTARY

Not entirely sure if it's worth adding reentrancy guards, but it can be done.

WRN-3	There is no processing of the value returned by the function
File	BaseParaSwapAdapter.sol
Severity	Warning
Status	Fixed at **fe05cecc**

DESCRIPTION

According to the standard `ERC-20` after a successful execution of operation, the function for working with tokens is returned to `true`.
You always need to check the value that returns the function after execution.
For this, there is even a special library `SafeERC20`.
But in the following lines it is not done:

- at the line BaseParaSwapAdapter.sol#L122

RECOMMENDATION

It is recommended for these operations to use the special library `SafeERC20`.

WRN-4	Ignored return value
File	BaseParaSwapAdapter.sol
Severity	Warning
Status	Fixed at **5b45be6a**

DESCRIPTION

At the line:

- BaseParaSwapAdapter.sol#L102
  the return value is ignored.
  But inside implementation POOL can implement surprised fee charging or withdrawing other amount in some specific case (as it is now).

RECOMMENDATION

It is recommended to add the check that the return value matches the expected.
For example add `minWithdrawAmount` argument and add `require(withdrawn >= minWithdrawAmount` .

CLIENT'S COMMENTARY

The lending pool doesn't charge a fee, it always returns the value passed in (unless it's MAX). We can add a require that the amount returned is exactly what was expected.

WRN-5	Pontential overflow
File	BaseParaSwapSellAdapter.sol
Severity	Warning
Status	Fixed at d26b1beb

DESCRIPTION

At the line:

- BaseParaSwapSellAdapter.sol#L53
  It is very easy to do overflow, just create a fake token with 100 decimals.

RECOMMENDATION

It is recommended to use SafeMath everywhere.

CLIENT'S COMMENTARY

There is no safe exponentiation function provided. I would say the _getDecimals function can require that the number of decimals is low enough that 10 ** decimals won't overflow.

2.4 COMMENTS

CMT-1	Unclear low-level calls
File	BaseParaSwapSellAdapter.sol
Severity	Comment
Status	**Fixed** at **11d0367d**

DESCRIPTION

At the lines:

- BaseParaSwapSellAdapter.sol#L68-L74
  some not really clear and intuitive low-level operations happen.

RECOMMENDATION

It is recommended to add comprehensive comments.

CMT-2	Using error constants
File	BaseParaSwapSellAdapter.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At the line

- BaseParaSwapSellAdapter.sol#L57
  is better off using error constants inside require.

RECOMMENDATION

It is recommended to use

```
library Errors {
  //common errors
  ...

  string public constant MIN_AMOUNT_EXCEEDS_MAX_SLIPPAGE = '1';
  ...
}
```

from the library: Errors.sol#L22-L118.

CLIENT'S COMMENTARY

This is not a core protocol contract so it should not use the Errors library (and no new errors should be added to this). I think the existing descriptive error messages are fine.

CMT-3	Using constants in substraction
File	BaseParaSwapSellAdapter.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At the line

- BaseParaSwapSellAdapter.sol#L55
  there is not optimized code.

RECOMMENDATION

It is recommended to use a constant variable containing the value of the expression:
`PercentageMath.PERCENTAGE_FACTOR - MAX_SLIPPAGE_PERCENT`.

CLIENT'S COMMENTARY

There is no need to change this, the optimizer can already see that both are constants and put the resulting constant in the code.

CMT-4	Function name is not suitable enough
File	BaseParaSwapAdapter.sol
Severity	Comment
Status	**Fixed** at **b13a01d8**

DESCRIPTION

At the line
BaseParaSwapAdapter.sol#L79
the `_pullAToken()` function, not only pulls Atoken, but also withdraws reserve from the `LENDING_POOL`.

RECOMMENDATION

It is recommended to change the function name to a more appropriate one.
For example `_pullAtokenThenWithdraw()`.

CMT-5	Dangerous strict equality
File	BaseParaSwapSellAdapter.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At the line

- BaseParaSwapSellAdapter.sol#L84
  there is a dangerous stric equality.

RECOMMENDATION

It is recommended to use delta expression.

CLIENT'S COMMENTARY

This is not dangerous, it is required for augustus to swap the whole input amount otherwise it will be left in the contract, so an exact comparison is correct.

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