SUSHISWAP BENTOBOX SMART CONTRACT AUDIT

February 15, 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 Sushiswap . 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

BentoBox is a full fledged lending platform which features:

- Isolated lending pairs. Anyone can create a pair, it's up to users which pairs they find safe enough. Risk is isolated to just that pair.
- Flexible oracles, both on-chain and off-chain.
- Liquid interest rates based on a specific target utilization, such as 75%.
- Contracts optimized for low gas.
- The supplied assets can be used for flash loans, providing extra revenue for suppliers.

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 inspected volume includes a set of smart contracts that are part of a new platform that allows users to deposit assets as collateral and borrow other assets against it. The developed functionality differs from the competitors' one. It adds new features for working with isolated credit pairs, flexible oracles and flash loans.

1.5 PROJECT DASHBOARD

Client	Sushiswap
Audit name	Bentobox
Initial version	c2e150b16b8764ebfe2e1e6e267ae14e10738065 2a67dd809af4f9206cfd5bd5018c67167d2f4262
Final version	2a67dd809af4f9206cfd5bd5018c67167d2f4262
SLOC	892
Date	2020-12-21 - 2021-02-15
Auditors engaged	2 auditors

FILES LISTING

BentoBox.sol	BentoBox.sol
LendingPair.sol	LendingPair.sol
ERC20.sol	ERC20.sol
Ownable.sol	Ownable.sol
SushiSwapSwapper.sol	SushiSwapSwapper.sol
ChainlinkOracle.sol	ChainlinkOracle.sol
PeggedOracle.sol	PeggedOracle.sol
CompositeOracle.sol	CompositeOracle.sol
SimpleSLPTWAP0Oracle.sol	SimpleSLPTWAP0Oracle.sol
CompoundOracle.sol	CompoundOracle.sol
SimpleSLPTWAP1Oracle.sol	SimpleSLPTWAP1Oracle.sol
BoringMath.sol	BoringMath.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	1
Warning	4
Comment	6

CONCLUSION

Smart contracts have been audited and several suspicious places have been spotted. During the audit no critical issues were found, one issue was marked as major because it could lead to some undesired behavior, also several warnings and comments were found and discussed with the client. After working on the reported findings all of them were resolved or acknowledged (if the problem was not critical).

2.FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

MJR-1	Incorrect events parameter
File	LendingPair.sol
Severity	Major
Status	Fixed at 2a67dd80

DESCRIPTION

At the lines below:

- LendingPair.sol#L252
- LendingPair.sol#L267
- LendingPair.sol#L282
- LendingPair.sol#L291
- LendingPair.sol#L306
- LendingPair.sol#L321

there are places where we have events which require an affected user address as a parameter, however in these cases `msg.sender` is wrongly used as a parameter. These functions accept special `user` parameter that should be used in events instead of `msg.sender`. The issue marked as major since it can fatally affect the user's side code that is based on events.

RECOMMENDATION

We suggest replacing `msg.sender` to `user`.

CLIENT'S COMMENTARY

> Events and functions have changed a fair bit, review of every event and the parameters is now part of our internal audit checklist.

> – 👤 auditor

> More parameters were added to events.

2.3 WARNING

WRN-1	No validation of the address parameter value in contract constructor
File	BentoBox.sol LendingPair.sol SushiSwapSwapper.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 line BentoBox.sol#L46 the `WETH` variable is set to the value of the `WETH_` input parameter.
- At line LendingPair.sol#L123 the `bentoBox` variable is set to the value of the `bentoBox__` input parameter.
- At line SushiSwapSwapper.sol#L17 the `bentoBox` variable is set to the value of the `bentoBox_` input parameter.
- At line SushiSwapSwapper.sol#L18 the `factory` variable is set to the value of the `factory_` input parameter.

RECOMMENDATION

In all the cases, it is necessary to add a check of the input parameter to zero before initializing the variables.

CLIENT'S COMMENTARY

> This is by design. This check would only benefit the developer/deployer or anyone who clones this. We tend to only add checks that improve security but we are keen to discuss this practice.

WRN-2	Loss of tokens is possible when sent to a zero address
File	ERC20.sol BentoBox.sol
Severity	Warning
Status	Fixed at 2a67dd80

DESCRIPTION

In smart contracts, tokens are transferred from one address to another and an approval is issued for such operations.
When sending tokens to a zero address, they will no longer be used and they will be lost.
Such actions are performed on the following lines:
In ERC20.sol on lines 22, 31-33, 39.
In BentoBox.sol on lines 107, 122, 126, 133, 161, 179.

RECOMMENDATION

Add address verification to zero.

CLIENT'S COMMENTARY

> Agreed. We added these checks to Transfer and TransferFrom - while this may technically break from the ERC20 standard and we normally don't like lots of checks, sending tokens to 0 by accident is common enough to warrant the extra gas for the check.

> – 👤 auditor
> Partialy fixed for ERC20 at 2a67dd80.
> The project uses ERC20.sol where these checks exist (except approve where it's not an issue). There is still no check at BentoBoxPlus.sol#L291.

WRN-3	It is possible to process a non-existing array element or skip an array element
File	LendingPair.sol BentoBox.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

- At line LendingPair.sol#L467 we are working with the elements of the `borrowFractions` array in a loop.
  For each element of the `users` array, there must be an element of the `borrowFractions` array.
  But if an error is made when transferring data for these arrays, then it is possible to refer to a nonexistent element of the array, or vice versa, any element will not be processed.
  This will cause the `liquidate()` function to work incorrectly.

- At line BentoBox.sol#L122 we are working with the elements of the `amounts` array in a loop.
  For each element of the `tos` array, there must be an element of the `amounts` array.
  But if an error is made when transferring data for these arrays, then it is possible to refer to a nonexistent element of the array, or vice versa, any element will not be processed.
  This will cause the `transferMultipleFrom()` function to work incorrectly.

RECOMMENDATION

Add a condition so that the length of the `users` array were equal to the length of the `borrowFractions` array.

CLIENT'S COMMENTARY

> This is by design, but we would love to discuss this and understand the best practices here and reasoning. In my testing Solidity throws an invalid opcode revert when you try to access elements that are out of bounds.

> – 👤 auditor
> An error if the number of elements in the second array is greater than the number of elements in the first array will be unnoticed.
> It is good programming practice to conduct checks.
> Additional gas will not be consumed for this.

WRN-4	Division by zero is possible
File	LendingPair.sol BentoHelper.sol ChainlinkOracle.sol CompoundOracle.sol SimpleSLPTWAP0Oracle.sol SimpleSLPTWAP1Oracle.sol SushiSwapSwapper.sol
Severity	Warning
Status	Fixed at 2a67dd80

DESCRIPTION

At the lines below division by zero is possible:

- LendingPair.sol#L227, the variable `_totalBorrow.fraction` can be equal to zero.
- LendingPair.sol#L259 the variable `_totalAsset.amount` can be equal to zero.
- LendingPair.sol#L274 the variable `_totalBorrow.amount` can be equal to zero.
- LendingPair.sol#L300 the variable `_totalAsset.fraction` can be equal to zero.
- LendingPair.sol#L315 the variable `_totalBorrow.fraction` can be equal to zero.
- LendingPair.sol#L469 the variable `_totalBorrow.fraction` can be equal to zero.
- BentoHelper.sol#L67 the variable `info[i].totalAssetFraction` can be equal to zero.
- BentoHelper.sol#L70 the variable `info[i].totalBorrowFraction` can be equal to zero.
- ChainlinkOracle.sol#L29 the variable `priceC` can be equal to zero.
- ChainlinkOracle.sol#L29 the variable `decimals` can be equal to zero.
- CompoundOracle.sol#L49 the variable `division` and the value `_getPrice(collateralSymbol)` can be equal to zero.
- CompoundOracle.sol#L55 the variable `division` and the value `_peekPrice(collateralSymbol)` can be equal to zero.
- SimpleSLPTWAP0Oracle.sol#L62 the variable `timeElapsed` can be equal to zero.
- SimpleSLPTWAP0Oracle.sol#L83 the variable `timeElapsed` can be equal to zero.
- SimpleSLPTWAP1Oracle.sol#L61 the variable `timeElapsed` can be equal to zero.
- SimpleSLPTWAP1Oracle.sol#L82 the variable `timeElapsed` can be equal to zero.
- SushiSwapSwapper.sol#L25 the variable `denominator` can be equal to zero.
- SushiSwapSwapper.sol#L32 the variable `denominator` can be equal to zero.

RECOMMENDATION

We will redo the division operation using the SafeMath Library.

CLIENT'S COMMENTARY

> – auditor
> Not fixed everywhere. For example SushiSwapSwapper.sol#L26 or SushiSwapSwapper.sol#L33.

2.4 COMMENTS

CMT-1	Using "magic" numbers
File	ERC20.sol BentoBox.sol LendingPair.sol
Severity	Comment
Status	Fixed at **2a67dd80**

DESCRIPTION

The use in the source code of some unknown where taken values impair its understanding:

- At line ERC20.sol#L55 the value is `\x19\x01`.
- At line ERC20.sol#L57 the value is `0x6e71edae12b1b97f4d1f60370fef10105fa2faae0126114a169c64845d6126c9`.
- At line BentoBox.sol#L171 the value is `0x23b872dd`.
- At line BentoBox.sol#L186 the value is `0xa9059cbb`.
- At line LendingPair.sol#L586 the value is `0xa9059cbb`.
- At lines 178, 389, 398, 503, 546 LendingPair.sol the value is `1e5`.
- At lines 177, 195, 198, 203 LendingPair.sol the value is `1e18`.
- At lines 86, 89 LendingPair.sol the value is `0x95d89b41`.
- At lines 96, 99 LendingPair.sol the value is `0x06fdde03`.
- At line LendingPair.sol#L106 the value is `0x313ce567`.
- At line LendingPair.sol#L471 the value is `1e23`.

RECOMMENDATION

It is recommended that you create constants with meaningful names to use numeric values.

CLIENT'S COMMENTARY

> Our internal audit now includes an item to change any 'magic number' to a constant with a clear name and a comment if needed.

CMT-2	The function returns a public variable
File	LendingPair.sol
Severity	Comment
Status	**Fixed** at **2a67dd80**

DESCRIPTION

For the LendingPair.sol#L243 line, the `updateExchangeRate()` function returns a value.
Lines LendingPair.sol#L155 and LendingPair.sol#L457 call this function.
But the return value is not processed.
The `updateExchangeRate()` function changes the `exchangeRate` public variable. There is no need to return a public variable.

RECOMMENDATION

Change the logic of the `updateExchangeRate()` function so that it did not return a public variable.

CLIENT'S COMMENTARY

> Nice find! That was just wasting gas, removed.

CMT-3	The function returns a variable, but it is not processed
File	ISwapper.sol LendingPair.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At line ISwapper.sol#L12, the `swap()` function returns a variable of type `uint256`. But after calling this function, there is no processing of the received value. It is found in the following places:

- At line LendingPair.sol#L428
- At line LendingPair.sol#L498
- At line LendingPair.sol#L519.

RECOMMENDATION

Add return value handling or rewrite the function logic so that it did not return a variable.

CLIENT'S COMMENTARY

> This has changed in the current version.

CMT-4	Define `symbol` and `name` methods as `external`
File	LendingPair.sol
Severity	Comment
Status	**Fixed** at **2a67dd80**

DESCRIPTION

At the line LendingPair.sol#L85 and LendingPair.sol#L95 methods `symbol` and `name` which is expected to be used as `external` define as `public`.

RECOMMENDATION

Define them as `external` to prevent internal usage.

CLIENT'S COMMENTARY

> Yes! Reviewing visibility of every function/variable is part of our internal audit, going forward we should catch this.

CMT-5	Remove unnecessary comment
File	LendingPair.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At LendingPair.sol#L429 the comment

> // TODO: Reentrancy issue? Should we take a before and after balance?

it is not really needed because it seems there is no re-entrancy issue here.

RECOMMENDATION

Remove the comment or discuss the problem.

CLIENT'S COMMENTARY

> Checking for reentrancy is something we would love to learn more about.

CMT-6	Forward success status
File	LendingPair.sol
Severity	Comment
Status	Fixed at 2a67dd80

DESCRIPTION

At LendingPair.sol#L236 the `success` variable is got, but not returned at LendingPair.sol#L243.

RECOMMENDATION

It may be useful for a caller to know if oracle value was really got or the old value was used, so `return success, exchangeRate`.

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