BONDAPPETIT PROTOCOL SMART CONTRACT AUDIT

March 18, 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 BondAppetit. 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

The first DeFi protocol that connects real-world debt instruments with the Ethereum ecosystem.

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

Audited scope includes contract which are the part of protocol that issue stable coins collateralized by a different assets such as stable coins and real world assets. System can be separated to several modules:

- stable coin - module that operates with different collaterals and issues stable coins
- governance - module that provide governance mechanism managed by governance token
- "periphery" - meta-module that includes different helper contracts

1.5 PROJECT DASHBOARD

Client	BondAppetit
Audit name	BondAppetit Protocol
Initial version	88680691fe8d872c5fc26e9500d19cf7caaa9861 355180f0aca0b29d60d808f761052956b7a3a159
Final version	c131f5dacf02ff8b6008c4da7788b71d86b26427
SLOC	1402
Date	2021-02-11 - 2021-03-18
Auditors engaged	2 auditors

FILES LISTING

AgregateDepositaryBalanceView.sol	AgregateDepositaryBal...
StableTokenDepositaryBalanceView.sol	StableTokenDepositary...
AccessControl.sol	AccessControl.sol
OwnablePausable.sol	OwnablePausable.sol
CollateralMarket.sol	CollateralMarket.sol
Issuer.sol	Issuer.sol
StableToken.sol	StableToken.sol
Staking.sol	Staking.sol
Treasury.sol	Treasury.sol
Vesting.sol	Vesting.sol
Market.sol	Market.sol
Investment.sol	Investment.sol
VestingSplitter.sol	VestingSplitter.sol
Budget.sol	Budget.sol
ProfitSplitter.sol	ProfitSplitter.sol
UniswapMarketMaker.sol	UniswapMarketMaker.sol
Buyback.sol	Buyback.sol
RealAssetDepositaryBalanceView.sol	RealAssetDepositaryBa...
DepositorCollateral.sol	DepositorCollateral.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	8
Warning	10
Comment	5

CONCLUSION

Smart contracts have been audited and several suspicious places were found. During audit 8 major issues were identified as they could lead to some undesired behavior also several issues were marked as warning and comments. After working on audit report all issues were fixed or acknowledged(if issue is not critical or major) by client.

2. FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

MJR-1	Potential `safeApprove` blocking
File	Investment.sol Market.sol Buyback.sol ProfitSplitter.sol UniswapMarketMaker.sol
Severity	Major
Status	Fixed at 6fbe358e

DESCRIPTION

At several places, e.g. Investment.sol#L182 contract perform `safeApprove` before uniswap's function call, however in case if uniswap doesn't use full provided allowance that can lead to blocking next `safeApprove` call because `safeApprove` requires zero allowance.

Another lines with same issue:

- Market.sol#L248
- Buyback.sol#L125
- ProfitSplitter.sol#L195
- ProfitSplitter.sol#L204
- UniswapMarketMaker.sol#L116
- UniswapMarketMaker.sol#L124
- UniswapMarketMaker.sol#L125
- UniswapMarketMaker.sol#L151
- UniswapMarketMaker.sol#L152
- UniswapMarketMaker.sol#L181

RECOMMENDATION

We recommend to always reset allowance to zero by calling `safeApprove` with `0` amount.

MJR-2	Wrongly calculated ETH amount to transfer
File	ProfitSplitter.sol
Severity	Major
Status	Fixed at 6fbe358e

DESCRIPTION

At lines ProfitSplitter.sol#L198-L205 contract swaps whole `splitterIncomingBalance` to ETH if `splitterIncomingBalance` insufficient to cover gap between `splitterEthBalance` and `amount`, in other words contract try to get as much as closer to `amount` ETH amount. However as we can see in this block of code contract assigns `amountsOut[1]` to `amount`, it's wrong because we need to assign `splitterEthBalance.add(amountsOut[1])`

RECOMMENDATION

We recommend to assign `splitterEthBalance.add(amountsOut[1])` to `amount` instead of `amountsOut[1]`

MJR-3	Potential re-entrancy problem
File	ProfitSplitter.sol
Severity	Major
Status	Fixed at 6fbe358e

DESCRIPTION

At the line ProfitSplitter.sol#L227 contract transfers incoming tokens to `recipient`, however that place can be re-entered in case of callbacks from `incoming` contract.

RECOMMENDATION

We recommend to add re-entrancy guard

MJR-4	Blocked LP tokens on contract
File	UniswapMarketMaker.sol
Severity	Major
Status	Fixed at 6fbe358e

DESCRIPTION

At the line UniswapMarketMaker.sol#L85 contract changes `incoming` token to another one, while transferring contract sends all remaining `incoming` tokens to `_recipient`, but contract never check remaining incoming <> support LP tokens on contract side. That tokens cannot be rescued anymore after changing incoming.

RECOMMENDATION

We recommend to remove all liquidity before changing `incoming` token

MJR-5	Missed depositary check
File	CollateralMarket.sol
Severity	Major
Status	Fixed at b57608a1

DESCRIPTION

In function `buy` defined at CollateralMarket.sol#L120 contract exchanges collateral tokens to stable tokens. But in case of wrong `depositary` that code will lead to collateralization disbalance, that is bad even you have manual `depositary` changing mechanism because `issuer` requires exact list of depositaries and transaction wont fail because `rebalance` call is fault tolerance.

RECOMMENDATION

We recommend check depositary

MJR-6	Invalid depositary add/remove logic
File	AgregateDepositaryBalanceView.sol
Severity	Major
Status	Fixed at 35a3f56d

DESCRIPTION

At lines AgregateDepositaryBalanceView.sol#L49, AgregateDepositaryBalanceView.sol#L62 are defined functions to add or remove depositaries, `depositariesIndex` map contains depositary indexes added to `depositaries` array. At line AgregateDepositaryBalanceView.sol#L50 contract requires that `depositariesIndex[depositary] == 0`, that check allow to add already added depositary that have `0` index. Same error at line AgregateDepositaryBalanceView.sol#L64 that don't allow to remove depositary that have index `0`

RECOMMENDATION

We recommend to remaster depositary existing check

MJR-7	Wrongly used `safeApprove`
File	Treasury.sol
Severity	Major
Status	Fixed at b57608a1

DESCRIPTION

At line Treasury.sol#L51 contract call `safeApprove` method, however that method fails if account have remaining allowed tokens.

RECOMMENDATION

We suggest to reset approval calling

```
ERC20(token).safeApprove(recipient, 0);
```

before setting new approval

MJR-8	Budget payment blocking
File	Budget.sol
Severity	Major
Status	Fixed at c131f5da

DESCRIPTION

In `pay` function of `Budget.sol` contract defined at Budget.sol#L109 contract sends ETH to recipients in loop using `transfer` method. As we know `transfer` method limited by 2300 gas, so any single recipient with payable fallback method can block whole `pay` function execution

RECOMMENDATION

We recommend to rework payment scheme to claimable model.

CLIENT'S COMMENTARY

This contract is used for disposition of funds to oracles, according to the list, approved by community. The possibility of using the bug is minimal, however we re-wrote the contract so that takeoff is made by the oracles.

2.3 WARNING

WRN-1	Potential integer overflow
File	Investment.sol
Severity	Warning
Status	Fixed at 6fbe358e

DESCRIPTION

At the line Investment.sol#L147 contract potentially can catch integer overflow in case if `cumulative.decimals() > 18`. Since `cumulative` token is not predefined contract we should check actual decimals amount

RECOMMENDATION

We recommend add check

WRN-2	Potential div by zero error
File	Market.sol
Severity	Warning
Status	Fixed at 6fbe358e

DESCRIPTION

At the line Market.sol#L189 contract can catch div by zero if `cumulativePrice` is zero.

RECOMMENDATION

We recommend add non-zero check

WRN-3	Vesting account duplication
File	VestingSplitter.sol
Severity	Warning
Status	Fixed at 6fbe358e

DESCRIPTION

At the line VestingSplitter.sol#L92 contract change vesting account, however input `accounts` array can contain duplicated accounts.

RECOMMENDATION

We recommend to introduce duplication check

WRN-4	Unchecked vesting contract address
File	VestingSplitter.sol
Severity	Warning
Status	Fixed at **6fbe358e**

DESCRIPTION

At the line VestingSplitter.sol#L111 contract accepts vesting contract address, but there is no sanity checks, so anyone can easily ask this contract to call another contract

RECOMMENDATION

We recommend add sanity check for vesting contract address

WRN-5	Wrong reward calculation of balance < 100
File	VestingSplitter.sol
Severity	Warning
Status	Fixed at 6fbe358e

DESCRIPTION

At the line VestingSplitter.sol#L126 contract calculate `reward` for account, however that calculation always return zero if `balance < 100`

RECOMMENDATION

We suggest to perform division after multiplication

WRN-6	Missed zero share check
File	ProfitSplitter.sol
Severity	Warning
Status	Fixed at 6fbe358e

DESCRIPTION

At the line ProfitSplitter.sol#L139 contract check that total shares sum including new share less or equal that 100, but never check that new share more that zero, so it's possible to add user with zero share.

RECOMMENDATION

We suggest to check that share more than 0.

WRN-7	Potential custodial asset collateral incorrect signatures
File	RealAssetDepositaryBalanceView.sol
Severity	Warning
Status	No issue

DESCRIPTION

This warning is about absent signature correctness checks in Proof data structure in RealAssetDepositaryBalanceView in here: RealAssetDepositaryBalanceView.sol#L88.

What kind of signatures are these? How do they get formed? Were they formed correctly and how to check that?

RECOMMENDATION

It is recommended to implement additional signature correctness checks, append comments about the nature of those signatures.

WRN-8	Mixed `msg.sender` and `_msgSender()`
File	Staking.sol StableToken.sol
Severity	Warning
Status	Fixed at 355180f0

DESCRIPTION

In some contracts used directly `msg.sender` instead of `_msgSender()`:

- Staking.sol#L173
- StableToken.sol#L12
- etc…

since OZ contract introduce Context based contract, all derived ones should use `_msgSender()`

RECOMMENDATION

We recommend to replace `msg.sender` to `_msgSender()`

WRN-9	Too flexible configuration
File	CollateralMarket.sol Issuer.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

Provided system have a list of contracts, some of them interact with each others. Contracts have too much implicit restrictions, e.g:

- CollateralMarket.sol requires that depositor should be listed in Issuer.sol
- Issuer.sol have methods to change list of depositors, so which means that in case of any changes depositor should be changes in CollateralMarket.sol at same time.
- Some contracts have flexible access list, that can lead to implicit wrong permissions

Too flexible and implicit configuration can lead to modules/contracts inconsistency. Moreover in some cases it could be fatal.

RECOMMENDATION

We suggest to strictly define possible invariants to reduce complexity.

WRN-10	Potentially wrong-sized access control list
File	AccessControl.sol
Severity	Warning
Status	Fixed at **b57608a1**

DESCRIPTION

This warning is about access list array being returned of a potentially wrong length in here: AccessControl.sol#L44.

It seems the actual purpose of this particular function is to provide a simple copy of the `allowed` array. It does not seem necessary to create a copy which length is bigger than the initial array.

RECOMMENDATION

It is recommended to provide a simple element-by-element array copy without implicit array size increase.

2.4 COMMENTS

CMT-1	Probably missed input check
File	Budget.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

In `transferETH` function of `Budget.sol` contract defined at Budget.sol#L61 contract sends ETH to `recipient` passed via arguments, however it seems `recipient` should be in `recipients` set, so it seems contract should check that before transfer.

RECOMMENDATION

We suggest to add particular check

CMT-2	Unneeded calculations
File	Market.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At the line Market.sol#L187 contract calculates `product` tokens amount, but line below contract recalculates this variable if `address(productToken) != currency`, so consequently first calculation unneeded because `productDecimals` and `tokenDecimals` are same if `productToken == currency`

RECOMMENDATION

We suggest to replace calculation with assignment `product = payment`

CMT-3	Total shares cache
File	ProfitSplitter.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At the line ProfitSplitter.sol#L126 contract calculate total shares sum, that information used when we adding new account and can be easily cached to save gas.

RECOMMENDATION

We recommend to cache current shares sum.

CMT-4	Potential collateralization imbalance
File	StableTokenDepositaryBalanceView.sol
Severity	Comment
Status	No issue

DESCRIPTION

In function `balance` defined at StableTokenDepositaryBalanceView.sol#L81 contract aggregates balances through different tokens, so function return sum of collateral assets. However, as we known price of some stable coins can be changed(especially algorithmic stable coins), so we can't simply calculate sum of tokens to get real assets value.

RECOMMENDATION

We recommend to use oracles to fetch real assets price

CLIENT'S COMMENTARY

There's no vulnerability here as we accept definite stable coins within this contract and they are assimilated 1:1 to our tokens.

CMT-5	Runtime-configured contract ownership
File	
Severity	Comment
Status	Acknowledged

DESCRIPTION

This comment is about very implicit runtime-configured contract ownership instead of explicit `Ownable`-alike constructions. Such an architecture makes the ownership deploy configuration-dependent, which is being hard to check after the deployment in comparison to simple code check.

RECOMMENDATION

It is recommended to either switch to the explicit ownership with `Ownable`, or to explicitly describe deployment params and the way to check them for everyone.

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