THRESHOLD-USD BAMM SECURITY AUDIT REPORT

February 13, 2024

MixBytes()

TABLE OF 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 the Client. 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 Security Assessment Methodology

A group of auditors are involved in the work on the audit. The security engineers check the provided source code independently of each other in accordance with the methodology described below:

1. Project architecture review:

- Project documentation review.
- General code review.
- Reverse research and study of the project architecture on the source code alone.

Stage goals
- Build an independent view of the project's architecture.
- Identifying logical flaws.

2. Checking the code in accordance with the vulnerabilities checklist:

- Manual code check for vulnerabilities listed on the Contractor's internal checklist. The Contractor's checklist is constantly updated based on the analysis of hacks, research, and audit of the clients' codes.
- Code check with the use of static analyzers (i.e Slither, Mythril, etc).

Stage goal

Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flash loan attacks etc.).

3. Checking the code for compliance with the desired security model:

- Detailed study of the project documentation.
- Examination of contracts tests.
- Examination of comments in code.
- Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit.
- Exploits PoC development with the use of such programs as Brownie and Hardhat.

Stage goal

Detect inconsistencies with the desired model.

4. Consolidation of the auditors' interim reports into one:

- Cross check: each auditor reviews the reports of the others.
- Discussion of the issues found by the auditors.
- Issuance of an interim audit report.

Stage goals

- Double-check all the found issues to make sure they are relevant and the determined threat level is correct.
- Provide the Client with an interim report.

5. Bug fixing & re-audit:

- The Client either fixes the issues or provides comments on the issues found by the auditors. Feedback from the Customer must be received on every issue/bug so that the Contractor can assign them a status (either "fixed" or "acknowledged").
- Upon completion of the bug fixing, the auditors double-check each fix and assign it a specific status, providing a proof link to the fix.
- A re-audited report is issued.

Stage goals

- Verify the fixed code version with all the recommendations and its statuses.
- Provide the Client with a re-audited report.

6. Final code verification and issuance of a public audit report:

- The Customer deploys the re-audited source code on the mainnet.
- The Contractor verifies the deployed code with the re-audited version and checks them for compliance.
- If the versions of the code match, the Contractor issues a public audit report.

Stage goals

- Conduct the final check of the code deployed on the mainnet.
- Provide the Customer with a public audit report.

Finding Severity breakdown

All vulnerabilities discovered during the audit are classified based on their potential severity and have the following classification:

Severity	Description
Critical	Bugs leading to assets theft, fund access locking, or any other loss of funds.
High	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss funds.
Low	Bugs that do not have a significant immediate impact and could be easily fixed.

Based on the feedback received from the Customer 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 Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future.

1.3 Project Overview

Threshold USD is a decentralized protocol that allows BTC and Ether holders to obtain maximum liquidity against their collateral without paying interest. After locking up BTC or ETH as collateral in a smart contract and creating an individual position called a "trove", the user can get instant liquidity by minting thUSD, a USD-pegged stablecoin. Each trove is required to be collateralized at a minimum of 110%. Any owner of thUSD can redeem their stablecoins for the underlying collateral at any time. The redemption mechanism along with algorithmically adjusted fees guarantee a minimum stablecoin value of USD 1.

1.4 Project Dashboard

Project Summary

Title	Description
Client	Threshold Network
Project name	B.Protocol
Timeline	May 15 2023 - December 29 2023
Number of Auditors	2

Project Log

Date	Commit Hash	Note
04.05.2023	800c6c19e44628dfda3cecaea6eedcb498bf0bf3	Commit for the audit
08.06.2023	d5e7a5202b4c28b5a825144f820d0b6e73ff1ceb	Commit for the re-audit
11.08.2023	2985371f6d1c0f12eaa262644002c0b0d96e76c4	Commit for the re-audit 2
29.12.2023	08b06a9f2a4f5eb23dcf9dbbf8c0d493921dcfdb	Commit for the deployment check

Project Scope

The audit covered the following files:

File name	Link
BAMM.sol	BAMM.sol

File name	Link
BLens.sol	BLens.sol
CropJoinAdapter.sol	CropJoinAdapter.sol
PriceFormula.sol	PriceFormula.sol
YieldBoxRebase.sol	YieldBoxRebase.sol

Deployments

Contract	Address	Comment
THUSDToken	0xCFC5bD...a38d29cf	
StabilityPool.sol	0xF6374A...96d06f29	tBTC collateral
PCV.sol	0x097f1e...71dd06cB	tBTC collateral
PriceFeed.sol	0x83aE39...56657a43	tBTC collateral
BAMM.sol	0x920623...1e834675	tBTC collateral
StabilityPool.sol	0xA18Ab4...48475a9f	ETH collateral
PCV.sol	0x1a4739...d07DD872	ETH collateral
PriceFeed.sol	0x684645...1Cdb732d	ETH collateral
BAMM.sol	0x1f4907...5d8B4DC5	ETH collateral

1.5 Summary of findings

Severity	# of Findings
Critical	0
High	0
Medium	4
Low	3

ID	Name	Severity	Status
M-1	The `swap` function doesn't check insufficient liquidity	Medium	Acknowledged
M-2	The `Trade` function doesn't use `minCollateralAmount`	Medium	Acknowledged
M-3	The `Receive` function revert	Medium	Fixed
M-4	Insuffisient quality of the `_collateralERC20` check in BAMM	Medium	Fixed
L-1	`Ownable` can be upgraded to `Ownable2Step`	Low	Acknowledged
L-2	`maxDiscount` is not customizable	Low	Acknowledged
L-3	Ignoring the `updatedAt` param	Low	Fixed

1.6 Conclusion

During the audit process 4 MEDIUM, and 3 LOW severity findings were spotted. After working on the reported findings, all of them were acknowledged or fixed by the client.

2.FINDINGS REPORT

2.1 Critical

Not Found

2.2 High

Not Found

2.3 Medium

M-1	The `swap` function doesn't check insufficient liquidity
Severity	Medium
Status	Acknowledged

Description

It's common practice to check that the protocol has enough amount of tokens to make a swap. The `minCollateralReturn` parameter has to be used only for preventing sandwich attacks (price manipulation) but in BAMM smart contract this parameter is also used as a liquidity check.

Let's suppose that an attacker is the biggest depositor of the BAMM contract. The attacker can always front-run swap function calls and keep the given `minCollateralReturn` collateral amount on the contract. Calling the BAMM.sol#L250 function with `minCollateralReturn` is a way to lose money even if the price has not been changed. As a result, users will always get `minCollateralReturn` tokens.

Recommendation

We recommend adding a lack of liquidity check.

Client's commentary

> In some cases user would need specify value that lower than 99.5% (if he profit from an arbitrage). So such check would not allow to do that. Besides this interface already used on B.Protocol side.

M-2	The `Trade` function doesn't use `minCollateralAmount`
Severity	Medium
Status	Acknowledged

Description

BAMM.sol#L268 can be used by mistake by someone and according to the M-1 it leads to losing money because the `minCollateralAmount` parameter is always zero.

Recommendation

We recommend adding a restriction that `trade` is called only by the Kyber protocol contracts.

Client's commentary

> This function is part of interface that planned to be used by bots

M-3	The `Receive` function revert
Severity	Medium
Status	Fixed in cf9b0db2

Description

The BAMM contract supports only one collateral token. If it is set to ERC20 tokens the BAMM.sol#L289 function has to revert all ETH transfers to the contract.

Recommendation

We recommend adding a check for the `collateralERC20` address and the sender's address.

M-4	Insuffisient quality of the `_collateralERC20` check in BAMM
Severity	Medium
Status	Fixed in 6399a825

Description

It is possible to provide the wrong address in the `_collateralERC20` parameter's value to the BAMM constructor:

- value that is different from StabilityPool,
- value that is not an ERC20 token,
- address of a token with the wrong decimals value (not 18)

Recommendation

We recommend adding a check of the `_collateralAddress` address parameter or removing this parameter from the constructor and getting it from the `StabilityPool` contract.

2.4 Low

L-1	`Ownable` can be upgraded to `Ownable2Step`
Severity	Low
Status	Acknowledged

Description

`BAMM` uses `Ownable`'s functionality:
BAMM.sol#L17.
The `Ownable` contract can be upgraded to OpenZeppelin's `Ownable2Step`:
Ownable2Step.sol.
`Ownable2Step` provides added safety due to its securely designed two-step process.

Recommendation

We recommend using `Ownable2Step`.

Client's commentary

> Owner will be DAO so everything related to this will be covered on governance side

L-2	`maxDiscount` is not customizable
Severity	Low
Status	Acknowledged

Description

`maxDiscount` doesn't have a setter to change the value.

- BAMM.sol#L34

Recommendation

We recommend adding a setter to manage this parameter similar to `setParams`.

Client's commentary

> This is part of B.Protocol so it will be set only once during deployment

L-3	Ignoring the `updatedAt` param
Severity	Low
Status	Fixed in 43c31714

Description

When `latestRoundData` is taken, the time of the last update is not checked. Thus, the Oracle's price may not be updated for a long time.

- BAMM.sol#L213

Recommendation

We recommend that you should not apply `compensateForTHUSDDeviation` if `Oracle` has not been updated for a long time.

3. ABOUT MIXBYTES

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

Contacts


https://github.com/mixbytes/audits_public


https://mixbytes.io/


hello@mixbytes.io


https://twitter.com/mixbytes