THRESHOLD-USD 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 collateralized stablecoin. A restricted list of highly liquid ERC20 tokens can be used as collateral. For sustainability, the stablecoin implements liquidation and redeem mechanisms. These mechanisms provide stability in a limited range. It should be noted that with a significant market collapse (more than 50%), with insufficient replenishment of the pledge by new users, the risk of unpeg is high.

1.4 Project Dashboard

Project Summary

Title	Description
Client	Threshold Network
Project name	Threshold USD
Timeline	April 10 2023 - December 29 2023
Number of Auditors	2

Project Log

Date	Commit Hash	Note
10.04.2023	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
ActivePool.sol	ActivePool.sol

File name	Link
BorrowerOperations.sol	BorrowerOperations.sol
CollSurplusPool.sol	CollSurplusPool.sol
DefaultPool.sol	DefaultPool.sol
GasPool.sol	GasPool.sol
HintHelpers.sol	HintHelpers.sol
Migrations.sol	Migrations.sol
MultiTroveGetter.sol	MultiTroveGetter.sol
PCV.sol	PCV.sol
PriceFeed.sol	PriceFeed.sol
SortedTroves.sol	SortedTroves.sol
StabilityPool.sol	StabilityPool.sol
THUSDToken.sol	THUSDToken.sol
TroveManager.sol	TroveManager.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	1
Medium	3
Low	5

ID	Name	Severity	Status
H-1	An attacker can steal the StabilityPool depositors profit	High	Acknowledged
M-1	An incorrect collateral value with decimals <> 18	Medium	Acknowledged
M-2	A cross-check of contract parameters	Medium	Fixed
M-3	Privileges granted to accounts as `system contracts` cannot be revoked	Medium	Fixed
L-1	Arbitrary transfer of `THUSD` by privileged accounts	Low	Fixed
L-2	MCP and CCR management	Low	Acknowledged
L-3	Mixed-up error messages in collateral assertions	Low	Acknowledged
L-4	A sanity check of the time interval	Low	Fixed
L-5	Magic values	Low	Fixed

1.6 Conclusion

During the audit process 1 HIGH, 3 MEDIUM, and 5 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

H-1	An attacker can steal the StabilityPool depositors profit
Severity	High
Status	Acknowledged

Description

The liquidation flow of the protocol is supposed to be as follows:

- users open troves and join `StabilityPool`
- anyone calls the `liquidateTroves` function that iterates the given troves and liquidates them one by one
- `StabilityPool` depositors move collateral gains to their troves and increase the `StabilityPool` ThUSD balance by getting more ThUSD.

By using a flash loan any user can bypass the provision of liquidity to the protocol for a long time and steal some of the `StabilityPool` provider's profit taking the following steps:

1. Let's wait for liquidation opportunities. The following notions are to be introduced: Lsum is the total liquidatable amount of ThUSD and FLusd is the amount of ThUSD that can be accumulated after depositing flashloaned collateral to the protocol; FLfee is the fees the attacker should pay for opening a trove, SPusd is the total ThUSD amount in `StabilityPool`.
   The conditions for an attack are:
   Lsum < SPusd + FLusd
   FLfee < Lsum * FLusd / SPusd
2. An attacker gets a flash loan and swaps the Lsum equivalent of the collateral token to ThUSD.
3. The attacker makes a deposit of all remaining collateral tokens to `StabilityPool`.
4. The attacker calls the TroveManager.sol#L464 function to liquidate the troves. If the CR system is lower than 150%, the amount of liqudated troves can be significantly bigger.

5. The attacker calls StabilityPool.sol#L283 then StabilityPool.sol#L310 of `StabilityPool` to move collateral to the attacker's trove. The CR System here has to be more than 150%.
   The attacker's trove here contains the initial collateral and collateral gain.
6. The attacker closes the trove providing the rest of ThUSD plus the Lsum equivalent from step 1.
7. The attacker returns the flash loan.

The attack's impact:

- loss of profit from liquidations by `StabilityPool` providers
- decreased motivation to use the Stability Pool which may cause Threshold USD to unpeg

Recommendation

We recommend that you use the time factor to prevent flash loan attacks.

Client's commentary

> The protocol still collects the 0.5% fee on the amount borrowed against the flashloan + the protocol still receives its portion of the liquidation.
> There are no incentives on the stability pool so most likely its only the protocol loan that will be in there.

2.3 Medium

M-1	An incorrect collateral value with decimals <> 18
Severity	Medium
Status	Acknowledged

Description

The PriceFeed.sol#L124 in `PriceFeed.sol` is not suitable for the `ERC20` tokens with `decimals`() not equal to 18. If the `decimals`() value is less than 18, the collateral value will be underestimated, which can result in a trove creation failure due to the BorrowerOperations.sol#L588 requirement. On the other hand, in the rare case where `decimals`() is greater than 18, the collateral value will be overestimated, which may allow minting of THUSD with an unreasonably low collateral value.

Recommendation

It is recommended that the `PriceFeed.sol` contract should rely on the actual `decimals`() value of the `ERC20` token to ensure correct calculations of the collateral value. If no `ERC20` tokens with `decimals`() other than 18 are planned to use, it is recommended to assert the `decimals`() of the `ERC20` token used in the contract is equal to 18.

Client's commentary

> thUSD will have two collaterals at the beginning: TBTC and ETH. Both of them are 18 decimals. In case we will consider to add another collateral with another decimals we will have to deploy new set of contracts anyway. And during this process we will adjust code to properly work with that.

M-2	A cross-check of contract parameters
Severity	Medium
Status	Fixed in dfba99bc

Description

If contracts `ActivePool`, `CollSurplusPool`, `DefaultPool`, `StabilityPool` and `BorrowerOperations` are deployed with the misconfigured `_collateralAddress` parameter, they will be unable to withdraw the deposited collateral.

Recommendation

It is recommended to enforce equality of the `_collateralAddress` parameter in the system contracts:

- ActivePool.sol#L43
- BorrowerOperations.sol#L97
- CollSurplusPool.sol#L49
- DefaultPool.sol#L48
- StabilityPool.sol#L206

M-3	Privileges granted to accounts as `system contracts` cannot be revoked
Severity	Medium
Status	Fixed in c7b29908

Description

Accounts can be marked as THUSDToken.sol#L58-L61. Such accounts have privileged access to some functionality: burn and arbitrary transfer between accounts. However, once granted, this privilege can't be revoked. If one of these accounts become compromised, there is no way to revoke its privileges.

Recommendation

It is recommended to introduce functions to pause (temporarily disable)`TroveManager`, `BorrowerOperations` and `StabilityPool` privileges for specified accounts.

2.4 Low

L-1	Arbitrary transfer of `THUSD` by privileged accounts
Severity	Low
Status	Fixed in e9d21765

Description

Accounts marked as THUSDToken.sol#L59 have a permission to perform transfers of any amount of `THUSD` from an arbitrary account to an arbitrary account by the THUSDToken.sol#L214 function. Additionally, THUSDToken.sol#L58-L59 can perform a similar arbitrary transfer by the THUSDToken.sol#L219 function. This function is designed to manage tokens in pools, however, it can transfer tokens between arbitrary accounts. Such undesired access is relatively safe, but can be involved into more complex attack scenarios.

Recommendation

It is recommended to improve the access control by disallowing arbitrary transfers.

L-2	MCP and CCR management
Severity	Low
Status	Acknowledged

Description

`MCP` and `CCR` are constants:
LiquityBase.sol#L21
Over time, the value and volatility of the collateral token may change. It will require to adapt `MCP` and `CCR` to new market behavior.

Recommendation

We recommend that you add methods for changing `MCP` and `CCR` parameters.

Client's commentary

> Independent of the collateral volatility, the 110% MCR shouldn't be changed, considering this is precisely the threshold ratio that maintains the peg.
> The CCR should be maintained as well, because it just serves to keep the TCR above 150%, keeping the system healthy with a high quantity of troves not close to the MCR.

L-3	Mixed-up error messages in collateral assertions
Severity	Low
Status	Acknowledged

Description

Assert messages in the `updateCollateralBalance` and `receive` functions are mixed-up (swapped).

```
require(collateralAddress != address(0),
    "ActivePool: collateral must be ETH");
```

```
require(collateralAddress == address(0),
    "ActivePool: collateral must be ERC20 token");
```

Although assertions are correct and generate revert() at proper conditions, the error messages are incorrect. It may cause an inaccurate diagnosis of the smart contract failures.

This issue appears at:

- ActivePool.sol#L151-161
- CollSurplusPool.sol#L117-125
- DefaultPool.sol#L113-122.

Recommendation

It is recommended to use error messages corresponding to the actual error.

Client's commentary

> In case when assertion `collateralAddress != address(0)` is wrong, that means that ETH should be used as collateral and wrong method was used.

L-4	A sanity check of the time interval
Severity	Low
Status	Fixed in a32b2f34

Description

The THUSDToken.sol#L79 parameter value has no sanity check. The deployer can unintentionally use a milliseconds value instead of a seconds value and produce undesired delays before applying timelocked actions.

Recommendation

It is recommended to perform a sanity check to disallow an unreasonable large parameter value.

L-5	Magic values
Severity	Low
Status	Fixed in 36c16ead

Description

Some code fragments use magic values to determine a trove status. It degrades the readability of the code:

- BorrowerOperations.sol#L508
- BorrowerOperations.sol#L513
- StabilityPool.sol#L651.

Recommendation

In favor of the code readability, it is recommended to use named constants instead of magic values.

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