# Security Assessment for xBacked

Report by Ulam Labs

# Security Assessment for xBacked

Findings and Recommendations Report Presented to:

## xBacked Team

May 13, 2022 Version: 0.1

Presented by:

Ulam Labs

Grabiszynska 163/502,

53-332 Wroclaw, POLAND

# Executive Summary

## Overview

xBacked engaged Ulam Labs to perform a Security Assessment for xBacked's **decentralized stablecoin smart contracts**.

The assessment was conducted remotely by the Ulam Labs Security Team. Testing took place on March 22 - May 17, 2022, and focused on the following objectives:

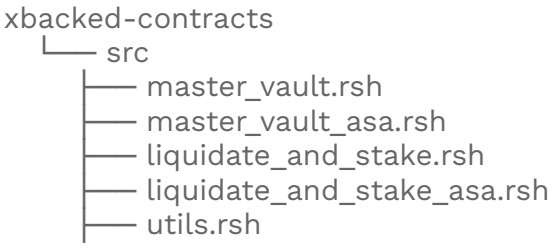
- Provide the customer with an assessment of their overall security posture and any risks discovered within the environment during the engagement.
- Provide a professional opinion on the maturity, adequacy, and efficiency of the security measures.
- Identify potential issues and include improvement recommendations based on the result of our tests.
- Confirmation of remediation for all reported issues.

This report summarizes the engagement, tests performed, and findings. It also contains detailed descriptions of the discovered vulnerabilities, steps the Ulam Labs Security Teams took to identify and validate each issue, and any applicable recommendations for remediation.

## Scope

The audit has been conducted on the commit c2ab218fc29356f977e0eb047f4007cff0ff4cf5 and a007383d7d46579fe0e8388326bb4fe23cded59c of xBacked private GitHub Repository.

Files included in the audit



## Key findings

During the Security Assessment for xBacked, we discovered the following findings.

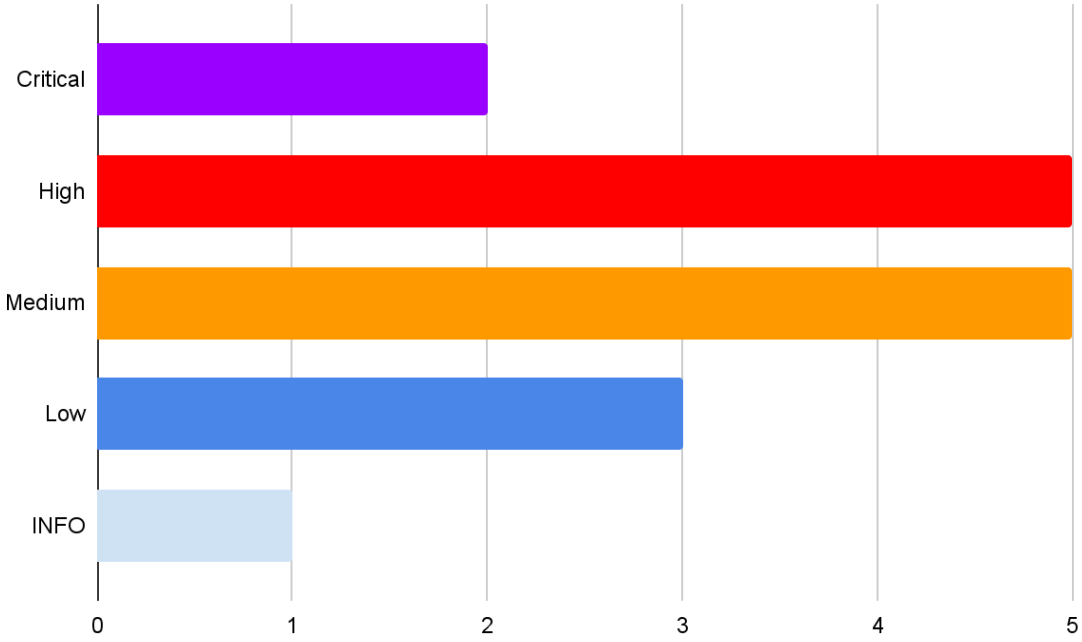



Chart 1: Findings by severity.

During the Security Assessment for xBacked, we discovered:

- 2 findings with a CRITICAL severity rating,
- 5 findings with a HIGH severity rating,
- 5 findings with a MEDIUM severity rating,
- 3 findings with a LOW severity rating,
- 1 findings with a INFO severity rating

All findings have been acknowledged and fixed by the xBacked team.

## Disclaimer

This report does not constitute legal or investment advice. The preparers of this report present it as an informational exercise documenting the due diligence involved in the secure development of the target contract only, and make no material claims or guarantees concerning the contract's operation post-deployment. The preparers of this report assume no liability for any and all potential consequences of the deployment or use of the contract.

Smart contracts are still a nascent software arena, and their deployment and public offering carries substantial risk. This report makes no claims that its analysis is fully comprehensive, and recommends always seeking multiple opinions and audits.

This report is also not comprehensive in scope, excluding a number of components critical to the correct operation of this system.

The possibility of human error in the manual review process is very real, and we recommend seeking multiple independent opinions on any claims which impact a large quantity of funds.

# Technical analysis & findings

## Redeem API call may stop working

Finding ID: XB-C1
Contract: master_vault
Severity: Critical
Status: Fixed

### Description

Redeem API call can be performed using one of two vaults stored in global state as **redeemableVaults**. Any vault can be proposed if its collateral ratio is less than one of the already proposed vaults. If one of the redeemable vaults is closed, it is impossible to calculate collateral radio, because of division by zero. In this case **pow(2, 64, 6)** was used, but it turned out to be evaluated as one by Reach.

### Impact

Collateral ratio has six decimals, so one is effectively **0.00001**. It is not possible to have a vault with such a ratio and a redeemable slot is lost forever. There are only two slots, so redeem API call can be blocked quickly. This is a critical problem, because the redeem API call is the only decentralized mechanism to restore target xUSD price if it is below **$1**. The only way to recover redeemable slots right now is to mark the contract as deprecated and deploy the new one.

### Solution

Use **UInt.max** instead of **pow(2, 64, 6)**.

### Status

Addressed by the xBacked team. The fix was applied to the source code and reviewed by Ulam Labs Security Team.

## Assets not having six decimals will not work as collateral

Finding ID: XB-C2
Contract: master_vault_asa
Severity: Critical
Status: Fixed

### Description

Contract is using **MICRO_UNITS** as the denominator in many equations, but it has six decimal places, which will not work for many popular assets (goBTC, goETH).

### Impact

The math is broken for asset based collateral if the collateral price has also six decimal places.

### Solution

Introduce a new parameter used as denominator instead of **MICRO_UNITS**.

### Status

Addressed by the xBacked team. The fix was applied to the source code and reviewed by Ulam Labs Security Team.

## Doubled interest while returning a debt

Finding ID: XB-H3
Contract: master_vault
Severity: High
Status: Fixed

### Description

Each API call altering the user vault must always update debt by adding interest accrued from last time it was updated. It is quite hard to achieve, because it requires calculations in many different places in the code. If the same procedure must be applied many times, it is easy to make a mistake, which was found in a return API call, where interest is added twice.

### Impact

Interest accrued is not as described in documentation. Users have to pay more than expected.

### Solution

User vault debt should be increased just once.

### Status

Addressed by the xBacked team. The fix was applied to the source code and reviewed by Ulam Labs Security Team.

## Vault can be liquidated even if its ratio never goes below the limit

Finding ID: XB-H4
Contract: master_vault
Severity: High
Status: Fixed

### Description

Vault can be liquidated if its ratio is below **LIQUIDATION_COLLATERAL_RATIO**, which is now **110%**. The vault can be liquidated until its ratio is above **MINIMUM_COLLATERAL_RATIO**, which is now **120%**. Only the liquidating flag is cleared. However in the drip interest API call, the liquidating flag can be set, even if it was unset before.

### Impact

It is unexpected behavior, which may cause financial loss if the vault is

liquidated.

### Solution

The drip interest API call, which is provided to update global interest accrued, should not be able to set liquidating flag, but just clear it.

### Status

Addressed by the xBacked team. The fix was applied to the source code and reviewed by Ulam Labs Security Team.

## Big vaults can be locked forever if not refreshed frequently

Finding ID: XB-H5
Contract: master_vault
Severity: High
Status: Open

### Description

Interest is proportional to the amount of time passed and vault debt. If multiplication of those two parameters causes integer overflow, no operations on such vault are possible. Currently the problem is mitigated, because vaults are refreshed every few days by bots. However bots can be attacked and then the problem may occur.

### Impact

If no operations are possible, there is no way to return, liquidate or redeem from such vault, causing huge financial loss for the vault owner.

### Solution

Avoid dangerous multiplication by using the **muldiv** function and introduce a max amount of time passed.

### Status

Addressed by the xBacked team. The fix was planned.

## Any vault can be proposed for Redemption

Finding ID: XB-H6
Contract: master_vault
Severity: High
Status: Open

### Description

If one of the slots of the redeemable vault is empty, any vault can be proposed. It is problematic, because proposer gets a **0.1%** of vault collateral as a reward, so it incentives proposer to choose the biggest vault, not the riskiest one. Proposers may prepare a sorted by collateral ratio list of vaults and propose them one by one in one block earning a lot from fees. The other problem is no collateral requirement. If a vault with minimal collateral is proposed, it is impossible to remove such vault from redeemable slots.

### Impact

Losing **0.1%** for the biggest vaults is quite a large amount of money and it is unexpected. On the other hand, vaults with low collateral are making redeem inefficient.

### Solution

Add requirements for minimum collateral level and remove the fee if the vault is safe.

### Status

Addressed by the xBacked team. The fixes were partially merged.

## Unaccounted for xUSD on the market possible if vault collateral ratio is below one

Finding ID: XB-H7
Contract: master_vault
Severity: High
Status: Open

### Description

If collateral price drops rapidly, it is possible that liquidation can make collateral even worse. In such a case it is possible to partially liquidate the vault and leave only a small amount of collateral (**0.000001** ALGO for example).

### Impact

All the debt left to repay is already on the market not backed by anything causing inflation.

### Solution

Such debt should be tracked and repaid using interest or fees.

### Status

Addressed by the xBacked team.

## All fees and interest can be burnt because of precision loss

Finding ID: XB-M8
Contract: master_vault
Severity: Medium
Status: Fixed

### Description

Fees and interest are transferred to DAO and liquidating contract using collect fees  or settle interest API call, half of available amount for each side. For both recipients, the amount to transfer is calculated by dividing all available tokens by two. Afterwards the total amount available is set to zero, causing one token to be lost forever.

### Impact

Losing one token normally is not a big problem, but in this case, if a malicious user keeps collecting fees or settles an interest API call frequently, no fees are distributed. Attack is easy to conduct and unstoppable. The only protection right now is blockchain transaction fees, which makes the attack expensive.

### Solution

There are two strategies to mitigate that problem. First is to store the remainder from dividing fees by two in a global state. Second, transferring funds to DAO without changes and everything that has been left to liquidating contract. Second option has been chosen.

### Status

Addressed by the xBacked team. The fix was applied to the source code and reviewed by Ulam Labs Security Team.

## Collateral price may be out of date if oracle is offline

Finding ID: XB-M9
Contract: master_vault
Severity: Medium
Status: Fixed

### Description

Contract behavior depends on collateral price heavily. Contract is not specifying any constraints regarding price and update interval. From a contract point of view, price can be anything greater than zero provided by a certain

address. It is extremely important to assure that oracle will be called frequently using data from many trusted sources and in case of a denial of service attack, the contract is secured. The oracle private key will probably be stored in the cloud, so there is the risk that it's going to be leaked.

### Impact

In the worst case, not fully backed xUSD are minted causing inflation. If the oracle private key is leaked, the attacker can set a minimal price locking all the collateral.

### Solution

Oracle implementation is out of scope of this document, but the contract must have a mechanism to be at least freezable on demand. Even better, but riskier because of false positives would be to freeze the contract automatically if a certain amount of time passed from the last price update. In order to recover from stolen oracle private key, a new API call updating oracle address should be introduced.

### Status

Addressed by the xBacked team. The fix was applied to the source code and reviewed by Ulam Labs Security Team.

## User vault may be overwritten

Finding ID: XB-M10
Contract: master_vault
Severity: Medium
Status: Fixed

### Description

It is possible to have no debt after the redeem API call is called on the vault. On the other hand, to create a vault, a deserialized local state must have debt equal to zero.

### Impact

If a create vault API call is possible on a vault containing some collateral, it is possible that the user accidentally will overwrite his local state and all the collateral will be lost.

### Solution

Checking if a vault is initialized shouldn't depend on vault debt, but on last update time, which is zero only for uninitialized vaults.

### Status

Addressed by the xBacked team. The fix was applied to the source code and reviewed by Ulam Labs Security Team.

## No anti slippage protection

Finding ID: XB-M11
Contract: master_vault
Severity: Medium
Status: Fixed

### Description

This is common practice to protect users against slippage, because in the dynamic environment like blockchain, things may change rapidly between presenting estimations and signing/sending transactions. Such protection is not guaranteed in XBacked contracts.

### Impact

Users can get unexpected results, because collateral price may change in the meantime..

### Solution

Adding minimum and maximum collateral price as input parameter.

### Status

Addressed by the xBacked team. The fix was applied to the source code and reviewed by Ulam Labs Security Team.

## Collateral ratio is calculated and stored, but not used anywhere

Finding ID: XB-I12
Contract: master_vault
Severity: Info
Status: Fixed

### Description

Collateral ratio is calculated and stored in user local state, but the only thing it provides is checking if vault debt is not zero (contract will panic if vault debt, divisor is zero).

### Impact

It takes blockchain space, it costs more while executing and makes code harder to read.

### Solution

Remove collateral ratio from local state scheme.

### Status

Addressed by the xBacked team. The fix was applied to the source code and reviewed by Ulam Labs Security Team.

## Integer overflow possible, while calculating proportion

Finding ID: XB-L13
Contract: master_vault
Severity: Low
Status: Fixed

### Description

In it common to calculate proportion in smart contracts using **A * B / C** formula. In Algorand, multiplication may cause integer overflow (which aborts contract execution), so its usage should be limited. Reach provides function **muldiv**, which stores result of **A * B** as two words, and then divide them by **C**. Such approach is followed most of the time, but not while calculating collateral to receive in redeem API call.

### Impact

If result of multiplication **MICRO_UNITS * amountOfXusdToRedeem** exceeds maximum **64** bit value, transaction is rejected.

### Solution

Use the **muldiv** function.

### Status

Addressed by the xBacked team. The fix was applied to the source code and reviewed by Ulam Labs Security Team.

## Precision loss while calculating interest rate

Finding ID: XB-L14
Contract: master_vault
Severity: Low
Status: Fixed

### Description

In order to calculate daily interest, vault debt must be multiplied by interest rate and amount of time passed and divided by amount of seconds in one year. Calculating the interest rate for one second is causing precision loss.

### Impact

Interest rate is smaller than specified in documentation.

### Solution

To avoid precision loss, division should always be the last operation.

### Status

Addressed by the xBacked team. The fix was applied to the source code and reviewed by Ulam Labs Security Team.

## One user can make contract exit unavailable

Finding ID: XB-L15
Contract: master_vault
Severity: High
Status: Accepted

### Description

Master vault contract does not support clear state operation. User with minimal collateral stored in his vault can clear his local state using clear state operation. Reach does not support such a case right now, so after destroying the user local state, the total collateral amount, stored in contract global state will never be zero.

### Impact

If the contract cannot be closed, it means all the xUSD replenished by admin and a small amount of network tokens are locked forever. This is not a big problem, because the admin has an almost infinite amount of xUSD and network tokens stored in contract are not significant.

### Solution

The only solution for this problem right now is to handle clear manually by writing custom code in teal. However it is even riskier than leaving it as it is, because reach manages global state in a tricky way and it is easy to make a mistake or clear program can become incompatible after Reach upgrade.

### Status

No fix is planned.

## Last consensus secs may be out of date

Finding ID: XB-I16
Contract: master_vault
Severity: Info
Status: Fixed

### Description

Contract is using the **lastConsensusTime** function, which takes time, when last consensus took time. It is taken from application global state and most of the time it is outdated

### Impact

Let’s assume we have one user and oracle is offline. If a user calls any API call and creates the vault at the same block, no interest will be accrued, even if debt is returned many years later.

### Solution

This is not a big program, because every price update will also update last consensus time. However it is important to remember that contract correctness depends on input frequency.

### Status

Acknowledged by the xBacked team. No fix required.

# Additional Recommendation

## Use multisig wallet for storing admin key

The account managing contract and xUSD assets should be a multisig wallet controlled by xBacked DAO. This can derisk scenarios when a centralized wallet is hacked, private key is leaked or key holder is coerced to hand over the key.

### Status

Acknowledged by xBacked team.

# Other

## Severity classification

We have adopted a severity classification inspired by the **Immunefi Vulnerability Severity Classification System - v2**. It can be found here.