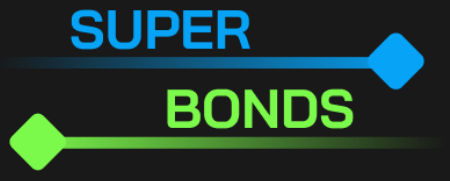
Security Assessment for



Report by Ulam Labs

Security Assessment for SuperBonds

Findings and Recommendations Report Presented to:

SuperBonds Team

March 31, 2022 Version: 0.1

Presented by:

Ulam Labs

Grabiszynska 163/502,

53-332 Wroclaw, POLAND

Executive Summary

Overview

SuperBonds engaged Ulam Labs to perform a Security Assessment for SuperBonds smart contracts.

The assessment was conducted remotely by the Ulam Labs Security Team. Testing took place on February 03 - March 29, 2022, and focused on the following objectives:

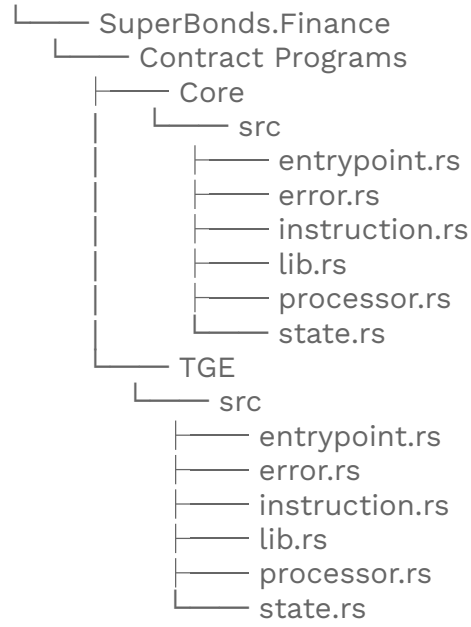
- Provide the customer with an assessment of their overall security posture and any risks discovered within the environment during the engagement.
- To provide a professional opinion on the maturity, adequacy, and efficiency of the security measures.
- To identify potential issues and include improvement recommendations based on the result of our tests.
- Confirmation of all remediations for 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 41b5bc462186570ca3eff13b3842dfee77f737d2 of SuperBonds private GitHub Repository. Subsequently, the codebase was prepared for public release, and the audited revision now bears the commit hash 1fb2e1e901f41b3281ab8f87ff30146e94fef57e.

Files included in the audit



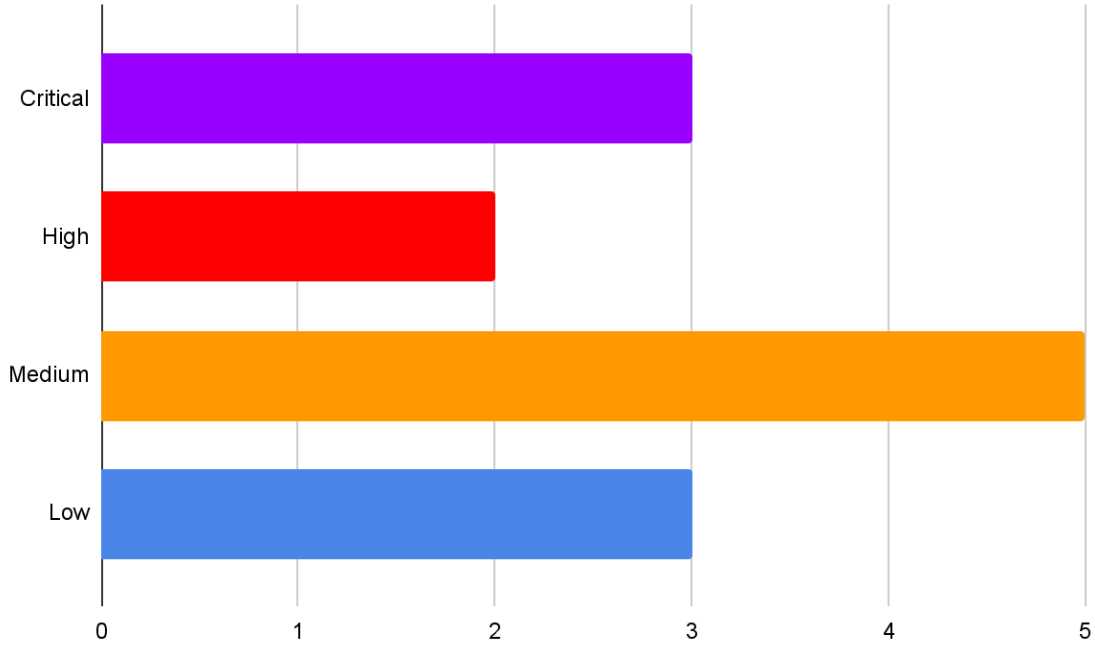
Key findings

During the Security Assessment for SuperBonds, we discovered:

- 3 findings with a CRITICAL severity rating.
- 2 findings with a HIGH severity rating.
- 5 findings with a MEDIUM severity rating.
- 3 findings with a LOW severity rating.

All findings have been acknowledged by the SuperBond team and fixed. Ulam Labs also reviewed the code after all fixes have been merged.

The following chart displays the findings by severity:



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

Multiple instances of staking pool are possible

Finding ID: SB-C27
Severity: Critical
Status: Fixed

Description

Core smart contract is designed with the assumption that there is only one account of instance **SB_Staking_State**, because the staking state key is not stored nor validated anywhere in code. It is extremely important to allow initialization of the staking pool only for governance authority. Unfortunately anyone is allowed to do it using following instructions:

- **process_trade**
- **process_redeem**
- **process_stake**
- **process_unstake**

Impact

Many instances of staking pool are breaking contract state causing minimal rewards for honest users and big rewards for users using one instance of staking pool for **trade/stake** and other for **redeem/unstake**.

Solution

Check if staking pool is initialized should be present in all the instructions, where staking pool account is used. It shall not be possible to initialize more than one instance of staking pool, and such action shall be possible only for governance authority.

Status

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

SB-C33 User is allowed to redeem using any pool

Finding ID: SB-C33
Severity: Critical
Status: Fixed

Description

Calling instruction **process_trade** is causing a new **TradeState** account to be initialized. Then it is possible to redeem the funds in **process_redeem instruction**, for users owning NFT, whose key is stored in a trade state account and using a pool, whose key is also stored there. However, the pool key is not validated.

Impact

Malicious users can choose one pool for trade and another for redeem, breaking contract state and earning more than expected.

Solution

Check if the trade state pool key is the same as the pool key should be added in **process_redeem** instruction.

Status

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

Any trade state may be overwritten

Finding ID: SB-C39
Severity: Critical
Status: Fixed

Description

In the **process_trade** instruction a new instance of **TradeState** is initialized. However, it is not checked if the account is already initialized.

Impact

Any user trade state may be overwritten and after that, the attacked user is unable to redeem funds anymore.

Solution

The check that the trade state is not initialized should be present.

Status

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

Last signer can set any farming account bypassing multisig protocol

Finding ID: SB-H28
Severity: High
Status: Fixed

Description

The most important parameters: admins, operator and governance are possible to change only using multisig protocol. At least three of them must accept **new_value**. However **rating_account** does not have to be accepted by all the admins, but it is passed by the last signer. A **farming_account** is a very important account holding 70% of all funds and it must be carefully validated

by all the admins, because it could be just delegated to the server authority, but actually owned by someone else.

Impact

The problem is a high severity issue, because all the benefits of setting operator accounts through multisig protocol are actually lost, because the last signer has a decisive impact on farming instructions safety.

Solution

There are few possible solutions for that problem.

1. Adding support for extra **data_type** and setting **farming_account** in separately.
2. Adding an extra field into **SB_MultiSig_Data** structure, so all the admins could verify a new farming account.
3. Adding a check if **farming_account** is owned by operator and **close_authority** is not set.

Status

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

SB-H29 Trading without fee possible

Finding ID: SB-H29
Severity: High
Status: Fixed

Description

SB fee is collected by burning some tokens from the user token account. It is important to check the token account mint, because a malicious user can burn his own worthless tokens.

Such check is present in almost all instructions except:

- **process_trade**

- **process_redeem**

Impact

Possibility to avoid fees is unexpected behavior causing financial loss for the contract owner.

Solution

SB mint id should be validated before burning tokens.

Status

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

SB-M34 SuperBonds_rate < 100 causes integer overflow

Finding ID: SB-M34
Severity: Medium
Status: Fixed

Description

Parameter **superBonds_rate** is a rate, which defines how much **bond_value** is going to be scaled. Parameter value should be between **0%** - **1000%**. If **superBonds_rate** is below **100%** calculated **bond_value** is smaller than unscaled value. Later on unscaled value is subtracted from scaled causing integer overflow.

Impact

Scenario, when superBonds_rate is below 100% is rather unlikely, because this parameter is set by governance authority. However, if it happens, any user will get 10% of all rewards for any trade.

Solution

Assert **superBonds_rate** is above **100%**.

Status

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

Updating reserved_multiplier_LP_Pool can cause data inconsistency or integer overflow

Finding ID: SB-M38
Severity: Medium
Status: Fixed

Description

Parameter **reserved_multiplier_LP_Pool** is modified in **process_update_parameter** instruction. Nothing else is changed with this parameter, but **adjustedLiquidity** is mostly based on that parameter.

Impact

In case, when reserved_multiplier_LP_Pool is changed, the remove liquidity instruction will calculate the amount to withdraw based on outdated **lp_price**. The other side effect is potential overflow if **reserved_multiplier_LP_Pool** is greater than before**.**

Solution

Update **lp_price** together with **reserved_multiplier_LP_Pool** in **process_update_parameter** and add a check if **reserved_multiplier_LP_Pool** is not causing integer overflow. As this action is performed in many places, it would be a great idea to extract such a procedure into a function.

Status

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

Parameter lp_price may change significantly while user is removing liquidity

Finding ID: SB-M42
Severity: Medium
Status: Fixed

Description

It is good practice to estimate how much the user will get (the worst case), before instruction to remove liquidity is called. In SB contract, there is a risk that **lp_price** may change significantly, causing smaller withdrawals than expected.

Impact

No slippage protection may lead to unexpected financial losses.

Solution

Add the instruction **min_amount** parameter and validate if the transfer amount is greater than that.

Status

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

Add anti slippage protection for bond yield and value at maturity calculation

Finding ID: SB-M43
Severity: Medium
Status: Fixed

Description

Right now the frontend is delivering estimates, what are expected **bond_yield** and **bond_value_at_maturity** values. However in the meantime the environment may change and different (possibly worse) values may be applied.

Impact

No slippage protection may lead to unexpected financial losses.

Solution

Add the instruction **min_interest** parameter and validate if the transfer amount is greater than that.

Status

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

Farming rewards can be distributed to any account

Finding ID: SB-M49
Severity: Medium
Status: Fixed

Description

There is no check if the account is, where farming rewards are going to be sent is actually owned by the recipient. The problem is even bigger, when the operator is passing such an account, because in the meantime the recipient account could change the ownership.

Impact

Rewards could be sent to the account, which is not controlled by the recipient anymore.

Solution

Validate if the destination account owner is the same as the trader.

Status

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

Reward distribution possible without updating trader state

Finding ID: SB-L23
Severity: Low
Status: Fixed

Description

Trader data owner is not validated, thus account with any data can be passed. Solana will not raise any problem while writing to the object, because it is possible to provide data, which is not causing any side effects.

Impact

If timestamps remain unchanged, a particular trader may receive reward many times for the same period. However, action can be triggered only by the operator, which is controlled by the contract owner.

Solution

Validate trader data account ownership before using its data.

Status

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

It is possible to transfer more tokens than required in process_deposit_trader_pool

Finding ID: SB-L48
Severity: Low
Status: Fixed

Description

In the case, when the **amount** sent by the operator is greater than **farming_amount**, the operator should send just **farming_amount** tokens.

Impact

If the **amount** is greater than **farming_amount**, the difference between them is lost. However both accounts are after all controlled by the contract owner, so nothing got lost permanently.

Solution

If the **amount** is greater than **farming_amount**, send **farming_amount** tokens.

Status

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

Risk factor can be updated without changing pool_risk_ factor_ vector

Finding ID: SB-L37
Severity: Low
Status: Fixed

Description

Vector **pool_risk_factor_vector** elements are calculated based on other parameters. One of those parameters is **risk_factor_x**. It is updated in **process_update_parameter** and **pool_risk_factor_vector** is not recalculated then.

Impact

Without manual **pool_risk_factor_vector** changing **risk_factor_x** has no effect.

Solution

Update **risk_factor_x** together with **pool_risk_factor_vector.**

Status

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

References

Severity classification

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