STATE MIND

Default Collateral

26-04-2024 - 01-05-2024

Table of contents

1. Project Brief

Title	Description
Client	Symbiotic
Project name	Default Collateral
Timeline	26-04-2024 - 01-05-2024
Initial commit	b620334d339a1ed6412b110cdcb6838e8df4519c
Final commit	e1892732cb6f5c57dfd40a7b3713642f92a993bb

Short Overview

Symbiotic is a shared security protocol enabling decentralized networks to control and customize their own multi-asset restaking implementation.

Collateral introduces a new type of asset that allows stakeholders to hold onto their funds and earn yield from them without needing to lock these funds in direct manner or convert them to another type of asset. Collateral represents an asset but does not require physically holding or locking this asset. The securities backing the Collateral can be in various forms, such as a liquidity pool position, some real-world asset, or generally any type of asset. Depending on the implementation of Collateral, this securing asset can be held within the Collateral itself or elsewhere.

Project Scope

The audit covered the following files:

- Factory.sol
- DefaultBond.sol
- DefaultBondFactory.sol
- Permit2Lib.sol

2. 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 to be transferred to any party.
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 of funds.
Informational	Bugs that do not have a significant immediate impact and could be easily fixed.

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

3. Summary of Findings

Severity	# of Findings
Critical	0 (0 fixed, 0 acknowledged)
High	0 (0 fixed, 0 acknowledged)
Medium	0 (0 fixed, 0 acknowledged)
Informational	4 (0 fixed, 4 acknowledged)
Total	4 (0 fixed, 4 acknowledged)

4. Conclusion

During the audit of the codebase, 4 issues were found in total:

- 4 informational severity issues (3 fixed, 1 acknowledged)

The final reviewed commit is e1892732cb6f5c57dfd40a7b3713642f92a993bb

5. Findings Report

INFORMATIONAL-01	Debt issuer and recipient can be the same address	Fixed at: cknowle

Description

Line: **DefaultBond.sol#L134**
The **issueDebt** function allows the same address to act as both the issuer and recipient of debt. It makes it possible to self-issue debt and artificially influence metrics **issuerRepaidDebt** and **recipientRepaidDebt** at the same time.

Recommendation

We recommend adding a check in the **issueDebt** function to prevent an address from being both issuer and recipient.

Client's comments

It is an intended behavior.

INFORMATIONAL-02	Factory doesn't emit event for new deployments	Fixed at: d8c3c01

Description

Line: **DefaultBondFactory.sol#L26**
The **DefaultBondFactory** does not emit an event when a new **DefaultBond** instance is deployed, making it harder to track off-chain.

Recommendation

We recommend adding an event emission in the **create** method to log each new deployment:

```
event BondCreated(address indexed bondAddress, address indexed asset);
```

INFORMATIONAL-03	Using immutable variable can save gas	Fixed at: d8c3c01

Description

Line: DefaultBondFactory.sol#L13
Considering that **BOND_IMPLEMENTATION** will never change, changing it to an immutable variable instead of a storage variable can save gas.

Recommendation

We recommend changing to immutable.

INFORMATIONAL-04	Unnecessary usage of ERC20	Fixed at: 34f28e7

Description

Lines:

- **DefaultBond.sol#L70**
- **DefaultBond.sol#L75**

The **DefaultBond.initialize()** function uses the **ERC20** contract to access basic token properties (**name**, **symbol**, and **decimals**). This creates an unnecessary dependency on the contract, while interface **IERC20Metadata.sol** works better for the task.

Recommendation

We recommend using the **IERC20Metadata** interface instead of the **ERC20** contract to reduce extra dependency.

STATE
MIND