BAIL security

Symbiotic

Bailsec Vanguard CISO
Pre Audit Report

FINAL REPORT

May ‘2025

Disclaimer:

Security assessment projects are time-boxed and often reliant on information that may be provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase.

The content of this assessment is not an investment. The information provided in this report is for general informational purposes only and is not intended as investment, legal, financial, regulatory, or tax advice. The report is based on a limited review of the materials and documentation provided at the time of the pre-audit, and the pre-audit results may not be complete or identify all possible vulnerabilities or issues. The pre-audit is provided on an "as-is," "where-is," and "as-available" basis, and the use of blockchain technology is subject to unknown risks and flaws.

The pre-audit does not constitute an endorsement of any particular project or team, and we make no warranties, expressed or implied, regarding the accuracy, reliability, completeness, or availability of the report, its content, or any associated services or products. We disclaim all warranties, including the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

We assume no responsibility for any product or service advertised or offered by a third party through the report, any open-source or third-party software, code, libraries, materials, or information linked to, called by, referenced by, or accessible through the report, its content, and the related services and products. We will not be liable for any loss or damages incurred as a result of the use or reliance on the pre-audit report or the smart contract.

The contract owner is responsible for making their own decisions based on the pre-audit report and should seek additional professional advice if needed. The pre-audit firm or individual assumes no liability for any loss or damages incurred as a result of the use or reliance on the pre-audit report or the smart contract. The contract owner agrees to indemnify and hold harmless the audit firm or individual from any and all claims, damages, expenses, or liabilities arising from the use or reliance on the pre-audit report or the smart contract.

By engaging in a smart contract pre-audit, the contract owner acknowledges and agrees to the terms of this disclaimer.

Bailsec Vanguard CISO

Bailsec will assign two senior security professionals to this engagement, providing ongoing support tailored to your project's needs.

Security requirements shift dramatically as a product evolves. In the early phases, strong architectural decisions lay the groundwork for safety and resilience. Once core features are locked in, establishing thorough test coverage becomes critical for maintaining quality and minimizing risk.

At every stage, having a neutral, expert voice helps ensure that core security principles are being properly applied. The Bailsec Vanguard CISO service offers exactly that, providing independent, high-impact guidance designed to enhance your protocol's defenses regardless of its maturity or complexity.

Why Choose Bailsec Vanguard CISO

Engaging an external security lead is a strategic move for teams aiming to become truly audit ready. Audit readiness goes far beyond having clean code. It involves proactively organizing your architecture, documentation, internal coordination, and team workflows in a way that enables effective, efficient, and thorough security reviews.

What the Vanguard CISO Delivers

Bailsec's Vanguard CISO service provides hands-on strategic direction across your development lifecycle. Whether refining your system architecture, advising on scalability and maintainability, or aligning engineering efforts with security best practices, we help teams make smart decisions early before problems become costly.

By embedding strong security fundamentals into the foundation of your product, you save time, avoid rework, and reduce the likelihood of critical vulnerabilities later in the process.

Conclusion

Security is not a layer you add at the end. It is a discipline that must inform decisions from day one. With Bailsec's Vanguard CISO, you gain access to dedicated, top-tier advisors who align security planning with your current development stage. This leads to smarter architectural choices, accelerated audit readiness, and more effective use of your security budget, ultimately helping your project scale securely and confidently.

1. Project Details

Important:
Please ensure that the deployed contract matches the source-code of the last commit hash.

Project	**Bailsec Vanguard CISO** - Symbiotic Pre Audit Report
Website	symbiotic.fi
Language	Solidity
Methods	Manual Analysis
Github repository	https://github.com/symbioticfi/middleware-sdk-mirror/tree/5c801a099f95591e676adb7bf3eb4cf17d83b063
Resolution 1	https://github.com/symbioticfi/middleware-sdk-mirror/tree/70114f97dc49cff884c067b8d080419c7bf38f38

2. Detection Overview

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)	Failed resolution
High	2	2			
Medium	2	1		1	
Low	3	1		2	
Informational	4	2		2	
Governance					
Total	11	6		5	

2.1 Detection Definitions

Severity	Description
High	The problem poses a significant threat to the confidentiality of a considerable number of users' sensitive data. It also has the potential to cause severe damage to the client's reputation or result in substantial financial losses for both the client and the affected users.
Medium	While medium level vulnerabilities may not be easy to exploit, they can still have a major impact on the execution of a smart contract. For instance, they may allow public access to critical functions, which could lead to serious consequences.
Low	Poses a very low-level risk to the project or users. Nevertheless the issue should be fixed immediately
Informational	Effects are small and do not post an immediate danger to the project or users
Governance	Governance privileges which can directly result in a loss of funds or other potential undesired behavior

3. Detection

Introduction

Between the 28th of May and 11th of June, Symbiotic engaged Bailsec to review the security of their new middleware SDK. Bailsec conducted a pre audit security review over 2 weeks with 2 engineers.

The first week was spent gaining a high level understanding of the core concepts in symbiotic as well as the start of manual review of the contracts.

In the second week we reviewed the codebase in depth and reported a total of 11 findings. Additionally we provided short risk assessments for the main features of the SDK, with the goal of protecting implementing users from potential pitfalls.

Overall the project follows good code practices, is well structured, and the architecture is designed to minimize risks. However due to the extensive size, some risks are still present.

Bailsec recommends Symbiotic to complete the following:

- extend unit testing
- implement integration testing
- carefully consider all external integration, even with symbiotic core
- conduct a security review after large codebase changes

Risk Assessment

Permissions

The permissions feature is a set of smart contracts that act as a wrapper for Openzeppelin's access control contracts.
These are modified to fit the overall architecture of Symbiotic, and allow for easier integration.
These contracts require careful consideration when implementing.

Users should make sure that:

- correct access control contracts are used for intended purpose
- all privileges are configured correctly for example the revoke role

Key Registries

The key registry is a smart contract deployed on the most secure chain, providing data on operator keys at specific points in time.

All smart contracts that are part of this feature are fairly simple, only providing set / get functions without complex logic.

When implementing with these users should make sure that:

- keys are fetched correctly using correct hints and time points
- the underlying hint system is not broken

Settlement

The settlement contract is an essential part of the Symbiotic SDK.
It implements a basic settlement flow for a decentralized network.
Settlement of the network state is done in epochs, where anyone can commit a valid network header (state).

This state has to be signed by the majority of validators and this is checked via ZK proof.
Settlement has 4 states, these include idle, commit, prolonged, and failed.

Users implementing settlement flows should consider:

- data about the current state should only be fetched from the settlement contract
- all permissioned functions should be examined carefully, they might lead to centralization risks
- it should be made sure which signature verifier system is being used and examine the security of it independently

Sig verifiers

The sig verifiers are part of the settlement flow, allowing to verify if the majority has voted on the submitted header.
There are different ZK verifier contracts, which are generated automatically by gnark.

Users integrating with the sig verifiers should make sure that:

- the ZK circuits are audited, they were not part of the scope
- correct verifiers are used to verify provided proofs

Voting Power Providers / Vault

The Self Register Voting Power Provider is currently the only provided implementation of the vault manager. This contract as the name suggests allows operators to self register and provides their voting power based on hints.
Hints are used to read data faster, read more in symbiotic core docs.

Users implementing these contracts should make sure that:

- Slashing and reward distribution is correctly implemented
- Access control is used when required
- Correct data is fetched using correct hints

Settlement/SettlementLogic

Issue_01	setEpochDuration does not include the checkPermission modifier
Severity	High
Description	The function setEpochDuration is a setter function that sets the duration of the epoch. However the function does not include any access control allowing anyone to change the value of the epoch duration. *function setEpochDuration(* *uint48 epochDuration* *) public virtual override {*
Recommendations	Consider adding the checkPermission modifier to the setEpochDuration function.
Comments / Resolution	Fixed, https://github.com/symbioticfi/middleware-sdk-mirror/commit/f498f825457a8958b944f343061c4859bb3213aa

Issue_02	Insufficient validation in setCommitDuration
Severity	**Low**
Description	in settlementLogic we are not allowed to set the epoch duration to be < the commit duration. *function setEpochDuration(* *uint48 epochDuration* *) public {* *if (epochDuration <= _getSettlementStorage()._commitDuration.latest()) {* *revert ISettlement.Settlement_EpochDurationTooShort();* *}* *EpochManagerLogic.setEpochDuration(epochDuration);* *}* However when setting the commitDuration, we do not check that the commitDuration is not > the epochDuration. Seeing as we want to ensure that epochDuration is always longer than commitDuration, it's important to add checks to ensure that the commitDuration cannot be longer than the epochDuration.
Recommendations	Consider adding a check to setCommitDuration that ensures that the commitDuration is not > epochDuration.
Comments / Resolution	Fixed, https://github.com/symbioticfi/middleware-sdk-mirror/commit/a0ad3a0e76434411dbcbeb11c40cc44cd6aae6ef

Issue_03	The setValSetHeader() function of SettlementLogic.sol should be marked as internal
Severity	**Low**
Description	*function setValSetHeader(* *ISettlement.ValSetHeader calldata header,* *ISettlement.ExtraData[] calldata extraData* *) public {* The setValSetHeader() function should be marked as internal. Because the function is located in a library this does not pose any immediate risks, but exposing this by incorrect implementation can lead to wrong settlement of network state, therefore this should be marked as internal.
Recommendations	Change visibility from public to internal
Comments / Resolution	Acknowledged, It is intended to decrease the bytecode size of the implementation contract.

Issue_04	inability to change prolongDuration
Severity	**Informational**
Description	*$._prolongDuration = settlementInitParams.prolongDuration;* *$._commitDuration.push(Time.timestamp(), settlementInitParams.commitDuration);* *$._requiredKeyTag.push(Time.timestamp(), settlementInitParams.requiredKeyTag);* *$._sigVerifier.push(Time.timestamp(), uint160(settlementInitParams.sigVerifier));* Unlike the other state variables, the prolongDuration does not have a setter and thus cannot be updated. Due to changing circumstances such as a change in epoch length, it's important to be able to update the prolongDuration.
Recommendations	Consider adding the ability to change the prolongDuration.
Comments / Resolution	Fixed, https://github.com/symbioticfi/middleware-sdk-mirror/commit/a579da08c85dd12d78767a575681a5ca7cbc45cf

Issue_05	Important setter functions do not protect against the 0 value
Severity	Informational
Description	Functions such as setCommitDuration, setRequiredKeyTag, setSigVerifier, and setGenesis do not have checks for the 0 value.
Recommendations	Consider adding 0 value checks to these functions.
Comments / Resolution	Fixed, https://github.com/symbioticfi/middleware-sdk-mirror/commit/707ca4907c6405bddbe9ad1e4fe31368eacd65cb

EpochManager/EpochManagerLogic

Issue_06	Incorrect epoch validation in setEpochDuration
Severity	**Medium**
Description	*function setEpochDuration(uint48 epochDuration, uint48 epochDurationTimestamp, uint48 epochDurationIndex) public {* *if (epochDurationIndex < getCurrentEpoch()) {* *revert IEpochManager.EpochManager_InvalidEpochDurationIndex();* *}* *setEpochDurationInternal(epochDuration, epochDurationTimestamp, epochDurationIndex);* *}* The function above sets the epochDuration to a new value, however the if statement incorrectly compares the epochDurationIndex with the currentEpoch using <. This allows changes in duration to the current epoch. Looking at the other function it can be noted that changing epochDuration should only change the duration of the subsequent epoch not the current epoch. *function setEpochDuration(* *uint48 epochDuration* *) public {* *setEpochDuration(epochDuration, getNextEpochStart(), getNextEpoch());* *}* as we can see the third argument which is epochDurationIndex is set to getNextEpoch.
Recommendations	Consider changing < to <= in the first comparison.
Comments / Resolution	Fixed, https://github.com/symbioticfi/middleware-sdk-mirror/commit/70114f97dc49cff884c067b8d080419c7bf38f38

Issue_07	_setEpochDuration is unused
Severity	Low
Description	The internal function _setEpochDuration is not called by any public function in EpochManager. *function _setEpochDuration(* *uint48 epochDuration,* *uint48 epochDurationTimestamp,* *uint48 epochDurationIndex* *) internal virtual {* *EpochManagerLogic.setEpochDuration(epochDuration, epochDurationTimestamp, epochDurationIndex);* *}*
Recommendations	Given that the function is not used, consider removing the function or adding a Public function which calls it.
Comments / Resolution	Acknowledged, The internal function inside the contract doesn't increase the bytecode size if not used; it is left here to be possible to use it conveniently

VaultManager/VaultManagerLogic

Issue_08	The Reward Distribution will fail due to lack of token approval
Severity	**High**
Description	the distributeStakerRewards() function of VaultManagerLogic.sol includes a call to the stakerRewards contract. This contract however transfers tokens from the sender to the contract. This transfer requires approval, which is not granted during the call: *function distributeStakerRewards(* *address stakerRewards,* *address token,* *uint256 amount,* *bytes memory data* *) public virtual override {* *IStakerRewards(stakerRewards).distributeRewards(NetworkManagerLogic.NETWORK(), token, amount, data);* *}* This will make every distributeStakerRewards call revert.
Recommendations	Add token approval.
Comments / Resolution	Fixed, https://github.com/symbioticfi/middleware-sdk-mirror/commit/b594d24195cbd35745008450a28d00156b12f523

Issue_09	epochDuration check is incorrect when comparing with slashingWindow
Severity	**Medium**
Description	in the internal function _validateVaultEpochDuration, there is a check that compares the slashingWindow with the vaultEpochDuration. *return slashingWindow <= vaultEpochDuration;* However the check is inclusive and this is flawed given that if the slashingWindow were to == vaultEpochDuration, we would now be in the next epoch not the current epoch. This can be observed by how the epoch is calculated *function currentEpoch() public view returns (uint256) {* *return (Time.timestamp() - epochDurationInit) / epochDuration;* *}* assuming we are in epoch 0, if the result of the timestamp-epochDurationInit is equal to epochDuration, we are considered to be in epoch 1 now instead of epoch 0.
Recommendations	Consider changing the check to not be inclusive.
Comments / Resolution	Acknowledged.

Issue_10	slashingWindow cannot be increased
Severity	Informational
Description	*function setSlashingWindow(* *uint48 slashingWindow* *) public {* *if (slashingWindow >= getSlashingWindow()) {* *revert IVaultManager.VaultManager_SlashingWindowTooLarge();* *}* *_getVaultManagerStorage()._slashingWindow = slashingWindow;* *}* The slashingWindow can only be decreased and never increased from the init value,
Recommendations	If this is intended behavior, acknowledge the issue or consider allowing slashingWindow to be increased.
Comments / Resolution	Acknowledged, It is an intended behavior.

OZAccessControl

Issue_11	The OZAccessControl.sol contract does not implement the by default public revokeRole function
Severity	**Informational**
Description	Openzeppelins access control contracts allow any user to revoke his role using the revoke function. Because the SDK is meant to override these functions, there is no good way for an implementation to override this on their own. This can lead to potential issues when for example implementing a blacklist.
Recommendations	Consider adding an override for revoke role that can be overridden.
Comments / Resolution	Acknowledged.