SALUS SECURITY

OCT 2023

CODE SECURITY ASSESSMENT

AMPHOR

Overview

Project Summary

- Name: Amphor
- Version: commit 9af4589
- Platform: EVM-compatible Chains
- Language: Solidity
- Repository:
    - https://github.com/mev-capital/amphor-core
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	Amphor
Version	v3
Type	Solidity
Date	Oct 9 2023
Logs	Sep 19 2023; Sep 27 2023; Oct 9 2023

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	3
Total informational issues	2
Total	5

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Potential issues in dependency management	Low	Configuration	Resolved
2	Missing event for critical state change	Low	Logging	Resolved
3	Centralization risk	Low	Centralization	Acknowledged
4	Gas optimization suggestions	Informational	Gas Efficiency	Resolved
5	Spelling errors	Informational	Code Quality	Resolved

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Potential issues in dependency management

Severity: Low	Category: Configuration
Target: - all	

Description

Instead of using a released version of OpenZeppelin library as a dependency, the Amphor codebase is using the code from commit 02ea017 of OpenZeppelin repo's master branch.

One issue with this dependency management approach is that code from the master branch includes unreleased features. The Amphor codebase has used two unreleased features: 1) Ownable supports setting the initial owner through its constructor, see PR 4267; 2) Rename rounding modes and complete with fourth, see PR 4455. It's not recommended to use unreleased features in OpenZeppelin because their security may not have undergone thorough auditing and testing.

Another issue with using code from the master branch is that later commits in the master branch could introduce breaking changes. One breaking change that will affect the Amphor codebase is PR 4551. This PR changes the import path for Pausable.sol, a file that Amphor depends on. Consequently, if the developer uses the `forge update lib/openzeppelin-contracts` command to update the OpenZeppelin library, `forge build` will fail to compile the project due to the changed import path for Pausable.sol.

Recommendation

We recommend using a released version of the OpenZeppelin library, as it is mature and stable. When installing dependencies with the `forge install` command, you can specify a particular version of the code by adding a version tag after the repository name. For instance, you can use `forge install OpenZeppelin/openzeppelin-contracts@v4.9.3` to install version 4.9.3 of OpenZeppelin library.

Status

The team has resolved this issue by using version 4.9.3 of OpenZeppelin library in commit ad6dd49.

2. Missing event for critical state change

Severity: Low	Category: Logging
Target: - src/AmphorSyntheticVault.sol	

Description

Important parameter or configuration changes should trigger an event to enable tracking off-chain.

However, AmphorSyntheticVault's setFees() function changes the feesInBps state variable - the performance fee parameter - but does not emit events.

Recommendation

Consider emitting an event for feesInBps change.

Status

This issue has been resolved by the team with commit d391e9f.

3. Centralization risk

Severity: Low	Category: Centralization
Target: - All	

Description

There is a privilege owner account in the Amphor codebase. The owner of the vault is responsible for using the end() function to return assets to the vault and unlock the lock period.

If the owner is a plain EOA account, this can be worrisome and may pose a risk to the users. Should the owner's private key be compromised, an attacker can transfer the owner role to an uncontrolled address, and call start() to withdraw the assets without calling end() to return them. As a result, users will lose their deposited assets.

Recommendation

We recommend transferring the owner role to a multi-sig account with timelock governors for enhanced security.

Status

This issue has been acknowledged by the team.

2.3 Informational Findings

4. Gas optimization suggestions

Severity: Informational	Category: Gas Efficiency
Target: - src/AmphorSyntheticVault.sol - src/AmphorSyntheticVaultWithPermit.sol	

Description

There are some instances throughout the codebase where changes can be made to improve gas consumption. For example:

- The _pause() internal function is modified by the whenNotPaused modifier, so the whenNotPaused modifier in the pause() function is redundant and can be removed to save gas. Similarly, the whenPaused modifier can be removed from the unpause() function.
- public functions might consume more gas than external functions. This is because with public functions, the EVM copies inputs (especially dynamic-sized arrays) into memory, while it reads from calldata if the function is external, which is cheaper. The depositWithPermit() public function in the AmphorSyntheticVaultWithPermit contract can be declared as external to save gas.
- Performing calculations for values that will not change is suboptimal. The operation 10 ** decimalsOffset can be replaced by using an immutable state variable.

Recommendation

Consider applying the gas optimization suggestions.

Status

The team has removed the redundant modifiers for pause() and unpasue() in commit 6fa0b53.

5. Spelling errors

Severity: Informational	Category: Code Quality
Target: - src/AmphorSyntheticVault.sol	

Description

src/AmphorSyntheticVault.sol:L292

```
    * @notice If the vault is locked or paused, usre should not deposit thus, the maxDeposit is 0.
```

src/AmphorSyntheticVault.sol:L301

```
    * @notice If the vault is locked or paused, usre should not mint thus, the maxMint is 0.
```

The word "usre" is misspelled, and should be "user" instead.

Recommendation

Consider correcting the spelling mistakes.

Status

The team has resolved this issue with commit 4034429.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit 9af4589:

File	SHA-1 hash
src/AmphorSyntheticVault.sol	9eecc8d3102650e774b8c6b4cab095c935b163dc
src/AmphorSyntheticVaultWithPermit.sol	4d0c6a624d03bea0fdcffa62a99a36e9a065a455