SALUS SECURITY

DEC 2024

CODE SECURITY ASSESSMENT

ASTHERUS

Overview

Project Summary

- Name: Astherus - AsBNB
- Platform: EVM-compatible chains
- Language: Solidity
- Repository:
    - https://github.com/astherus-contract/ass-bnb-earn-contract
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	Astherus - AsBNB
Version	v2
Type	Solidity
Dates	Dec 11 2024
Logs	Nov 29 2024; Dec 02 2024; Dec 11 2024

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	2
Total Low-Severity issues	1
Total informational issues	4
Total	7

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 **4**
1.1 About SALUS 4
1.2 Audit Breakdown 4
1.3 Disclaimer 4
Findings **5**
2.1 Summary of Findings 5
2.2 Notable Findings 6
1. Wrong activity removal logic 6
2. The newly added activity may affect the originally active activity 8
3. Use safeTransfer()/safeTransferFrom() instead of transfer()/transferFrom() 9
2.3 Informational Findings 10
4. Incomplete initialization 10
5. Missing two-step transfer ownership pattern 11
6. Comment error 12
7. Use of floating pragma 13
Appendix **14**
Appendix 1 - Files in Scope 14

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	Wrong activity removal logic	Medium	Business Logic	Resolved
2	The newly added activity may affect the originally active activity	Medium	Business Logic	Resolved
3	Use safeTransfer()/safeTransferFrom() instead of transfer()/transferFrom()	Low	Risky External Calls	Resolved
4	Incomplete initialization	Informational	Code Quality	Resolved
5	Missing two-step transfer ownership pattern	Informational	Business Logic	Resolved
6	Comment error	Informational	Inconsistency	Resolved
7	Use of floating pragma	Informational	Configuration	Resolved

2.2 Notable Findings

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

1. Wrong activity removal logic

Severity: Medium	Category: Business Logic
Target: - src/YieldProxy.sol	

Description

The `endActivity()` function causes `lastUpdatedActivityIdx` to point to the `activity` that has been removed.

Assuming there are four activities with indexes 0, 1, 2, and 3, after settling the activity with index 0, `lastUpdatedActivityIdx` will become 1. At this point, if the `endActivity` function is called with a parameter of 1(The same applies to other parameters), the activity with index 1 will be successfully ended, but at the end of the function, `lastUpdatedActivityIdx` is set to 1. This will cause an accounting error in the contract. For example, if the `settleActivity()` function is called afterward, although the logic executes normally, it effectively settles an activity that has already been ended.

src/YieldProxy.sol:L266 - L281

```
function endActivity(uint256 numberOfActivity) external onlyRole(MANAGER) whenNotPaused
{
  ...
  for (uint256 i = idx; i < (lastUpdatedActivityIdx + numberOfActivity); i++) {
    ...
    activities[i].rewardedTime = block.timestamp;
    idx = i;
    emit ActivitySettled(block.timestamp, activities[idx].tokenName);
  }
  lastUpdatedActivityIdx = idx;
}
```

src/YieldProxy.sol:L177 - L204

```
function settleActivity() external onlyRole(BOT) whenNotPaused {
  ...
  // save rewarded time
  activities[lastUpdatedActivityIdx].rewardedTime = block.timestamp;
  // update last updated activity index
  ++lastUpdatedActivityIdx;
  ...
}
```

Recommendation

Consider modifying the `endActivity()` function to set `lastUpdatedActivityIdx` to the next activity after the one that has been ended.

Status

The team has resolved this issue in commit bb35f6f.

2. The newly added activity may affect the originally active activity

Severity: Medium	Category: Business Logic
Target: - src/YieldProxy.sol	

Description

The `activitiesOnGoing()` function only checks the tail element. When the tail element is not active, all activities will be considered non-active.

src/YieldProxy.sol:L126 - L133

```
function activitiesOnGoing() external view override returns (bool) {
  ...
  return
    activities[activities.length - 1].startTime <= block.timestamp &&
    activities[activities.length - 1].rewardedTime == 0;
}
```

When there are existing `activities`, the `manager` can add new `activities`. If the newly added activity has a `startTime` later than the currently active `activity`, due to only checking the tail of the queue, the originally active activity will not be considered active. It will only be considered active when the `startTime` of the newly added activity is reached. By that time, the originally active activity may already have reached its `endTime`.

Recommendation

Consider adding restrictions to the `addActivity()` function or modifying the judgment logic of the `activitiesOnGoing()` function to prevent newly added activities from impacting existing ones.

Status

The team has resolved this issue in commit bb35f6f.

3. Use safeTransfer()/safeTransferFrom() instead of transfer()/transferFrom()

Severity: Low	Category: Risky External Calls
Target: - src/AsBnbMinter.sol	

Description

src/AsBnbMinter.sol: L312-L339

```
function _mint(uint256 amountIn) private returns (uint256) {
    require(amountIn >= minMintAmount, "amount is less than minMintAmount");
    // transfer token to YieldProxy
    token.transferFrom(msg.sender, address(this), amountIn);
    token.safeIncreaseAllowance(yieldProxy, amountIn);
    IYieldProxy(yieldProxy).deposit(amountIn);
    ......
}
```

Tokens not compliant with the ERC20 specification could return false from the transfer function call to indicate the transfer fails, while the calling contract would not notice the failure if the return value is not checked. Checking the return value is a requirement, as written in the EIP-20 specification:

```
Callers MUST handle false from returns (bool success). Callers MUST NOT assume that
false is never returned!
```

Recommendation

Consider using the `SafeERC20` library implementation from OpenZeppelin and `safeTransferFrom` when transferring `ERC20` tokens.

Status

The team has resolved this issue in commit bb35f6f.

2.3 Informational Findings

4. Incomplete initialization

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

Description

The contracts inherit a total of four upgradeable contracts, but during initialization, only two of them were initialized. The `AccessControlUpgradeable` and `UUPSUpgradeable` contracts were not initialized.

Recommendation

It is recommended to add logic to initialise the remaining two contracts.

Status

The team has resolved this issue in commit bb35f6f.

5. Missing two-step transfer ownership pattern

Severity: Informational	Category: Business logic
Target: - src/AsBNB.sol	

Description

The `AsBNB` contract inherits from the `Ownable` contract. This contract does not implement a two-step process for transferring ownership. Thus, ownership of the contract can easily be lost when making a mistake in transferring ownership.

Recommendation

Consider using the Ownable2Step contract from OpenZeppelin instead.

Status

The team has resolved this issue in commit bb35f6f.

6. Comment error

Severity: Informational	Category: Inconsistency
Target: - src/AsBnbMinter.sol	

Description

`AsBnb` tokens should be freely transferred by users, and they should have the right to burn them to retrieve the original tokens. But the comment indicates that only the `yieldProxy` has the right to do so.

src/AsBnbMinter.sol:L165 - L169

```
/**
 * @dev burn asBnb - only yieldProxy can call this function
 * @param amountToBurn - amount of asBnb to burn
 */
function burnAsBnb(uint256 amountToBurn) external override whenNotPaused nonReentrant
returns (uint256) {
```

Recommendation

Consider revising the comment content.

Status

The team has resolved this issue in commit bb35f6f.

7. Use of floating pragma

Severity: Informational	Category: Configuration
Target: - All	

Description

```
pragma solidity ^0.8.20;
```

All contracts use a floating compiler version `^0.8.20`.

Using a floating pragma `^0.8.20` statement is discouraged, as code may compile to different bytecodes with different compiler versions. Use a locked pragma statement to get a deterministic bytecode. Also use the latest Solidity version to get all the compiler features, bug fixes and optimizations.

Recommendation

It is recommended to use a locked Solidity version throughout the project. It is also recommended to use the most stable and up-to-date version.

Status

The team has resolved this issue in commit bb35f6f.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit e1561e3:

File	SHA-1 hash
src/AsBNB.sol	05266a2d8ee507dc8e83d40fffe73a9c812a736e
src/AsBnbMinter.sol	90767278443c57fff6942b0925eb4f8e1c175a7e
src/Timelock.sol	71a42a8346b50c56100a9759f2963488d344f5b4
src/YieldProxy.sol	289c9c42c8d2eab39919c513b454aaa3c6a25cff
src/libraries/FullMath.sol	030310b38cd52d6ff89a05bd4993684ec6733be4