SALUS SECURITY

FEB 2025

CODE SECURITY ASSESSMENT

SYNTH-X

Overview

Project Summary

- Name: Synth - USD0x Pre Deposit
- Platform: EVM-compatible chains
- Language: Solidity
- Repository:
  - https://github.com/Synth-X/usd0x-pre-deposit-contracts
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	Synth - USD0x Pre Deposit
Version	v2
Type	Solidity
Dates	Feb 26 2025
Logs	Feb 26 2025; Feb 26 2025

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	1
Total Low-Severity issues	3
Total informational issues	1
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 **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. Centralization risk 6

2. Potential fallback risk for whitelisted users 7

3. Tokens with a value lower than the pre-sale price are not supported as assets 8

4. Missing events for functions that change critical state 9

2.3 Informational Findings 10

5. Missing two-step transfer ownership pattern 10

Appendix **11**

Appendix 1 - Files in Scope 11

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	Centralization risk	Medium	Centralization	Mitigated
2	Potential fallback risk for whitelisted users	Low	Business Logic	Resolved
3	Tokens with a value lower than the pre-sale price are not supported as assets	Low	Business Logic	Acknowledged
4	Missing events for functions that change critical state	Low	Logging	Resolved
5	Missing two-step transfer ownership pattern	Informational	Business logic	Resolved

2.2 Notable Findings

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

1. Centralization risk	
Severity: Medium	Category: Centralization
Target: - contracts/USD0XPreDeposit.sol	

Description

The `` `USD0XPreDeposit` `` contract has privileged accounts `` `owner` `` .

If `` `owner` ``'s private key is compromised, an attacker can perform a series of privileged operations, such as altering the `` `PreDepositStatus` ``, modifying the `` `USD0XToken` ``, adding or deleting `` `Assets` ``, and adding or removing `` `whitelisted` `` addresses.

If the privileged accounts are plain EOA accounts, this can be worrisome and pose a risk to the other users.

Recommendation

We recommend transferring privileged accounts to multi-sig accounts with timelock governors for enhanced security. This ensures that no single person has full control over the accounts and that any changes must be authorized by multiple parties.

Status

The team stated that they will transfer the owner address to a multi-sig account after deployment.

2. Potential fallback risk for whitelisted users	
Severity: Low	Category: Business logic
Target: - contracts/USD0XPreDeposit.sol	

Description

When there are multiple `asset`s in a contract, adding a whitelist with the `vault` address of only one `asset` for a particular user will result in an error when that user participates in the pre-sale with the other `asset`, and will not be able to transact with the normal user's `vault`.

Recommendation

It is recommended that the checking logic be changed as follows:

```
function preDeposit(address asset, uint256 amount) external nonReentrant {
    require(preDepositEnabled, "Pre-Deposit is not active");
    require(supportedAsset.contains(asset), "Asset not supported");

    AssetInfo storage info = assetInfo[asset];
    uint256 usd0xAmount = amount * info.conversionFactor;
    require(usd0xAmount >= minAmount, "Amount below minimum");

    address user = msg.sender;
    address vault = whitelistedVaults[user][asset] == address(0)? info.vault :
whitelistedVaults[user][asset];

    IERC20(asset).safeTransferFrom(user, vault, amount);
    usd0xBalances[user] += usd0xAmount;

    emit PreDeposited(user, asset, amount, usd0xAmount);
}
```

Status

The team has resolved this issue in commit 3ce3398.

3. Tokens with a value lower than the pre-sale price are not supported as assets	
Severity: Low	Category: Business logic
Target: - contracts/USD0XPreDeposit.sol	

Description

If the contract supports tokens with a value lower than the pre-sale price as assets, `` `preDeposit()` `` will not be able to handle this situation correctly. Suppose a user uses `` `Token A` `` to purchase `` `USD0x` ``, and the value of `` `Token A` `` is lower than the pre-sale price of `` `USD0x` ``. With the minimum `` `conversionFactor` `` set to 1, the user could buy `` `USD0x` `` at a lower price.

Recommendation

It is recommended to take this into consideration when adding assets or to modify the `` `usd0xAmount` `` calculation logic to support such lower-value tokens.

Status

This issue has been acknowledged by the team.

4. Missing events for functions that change critical state	
Severity: Low	Category: Logging
Target: - contracts/USD0XPreDeposit.sol	

Description

Events allow capturing the changed parameters so that off-chain tools/interfaces can register such changes that allow users to evaluate them. Missing events do not promote transparency and if such changes immediately affect users' perception of fairness or trustworthiness, they could exit the protocol causing a reduction in protocol users.

In the `` `USD0XPreDeposit1` `` contract, events are lacking in the privileged setter functions `` `updatePreDepositStatus()` ``, `` `enableClaim()` ``, `` `setUSD0XToken()` ``, `` `setMinAmount()` ``.

Recommendation

It is recommended to emit events for critical state changes.

Status

The team has resolved this issue in commit 8a50ce2.

2.3 Informational Findings

5. Missing two-step transfer ownership pattern	
Severity: Informational	Category: Business logic
Target: - contracts/USD0XPreDeposit.sol	

Description

The `USD0XPreDeposit1` contract inherit from the `OwnableUpgradeable` 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 Ownable2StepUpgradeable contract from OpenZeppelin instead.

Status

The team has resolved this issue in commit 3e44c8f.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit 0748d24:

File	SHA-1 hash
contracts/USD0XPreDeposit.sol	aa4c7e523133022fc6a32772dbc43ab820f42d26