SALUS SECURITY

DEC 2025

CODE SECURITY ASSESSMENT

SWAPX

Overview

Project Summary

- Name: SwapX - staking
- Platform: Xone
- Language: Solidity
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	SwapX - staking
Version	v2
Type	Solidity
Dates	Dec 03 2025
Logs	Dec 03 2025; Dec 03 2025

Vulnerability Summary

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

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. Incorrect stakeIndex emitted in Staked event 7

3. SupportsInterface breaks ERC165 compatibility 8

2.3 Informational Findings 9

4. Comment mismatch code in multiplier usage 9

Appendix **10**

Appendix 1 - Files in Scope 10

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	Acknowledged
2	Incorrect stakeIndex emitted in Staked event	Low	Logging	Acknowledged
3	SupportsInterface breaks ERC165 compatibility	Low	Configuration	Acknowledged
4	Comment mismatch code in multiplier usage	Informational	Inconsistency	Acknowledged

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/StakingPool.sol	

Description

There is a privileged owner role in the StakingPool contract, the owner can upgrade the implementation via `` `_authorizeUpgrade()` `` and to withdraw all funds from the contract via `` `emergencyWithdraw()` ``.

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

contracts/StakingPool.sol:L182-L186

```
function emergencyWithdraw() external onlyOwner {
    uint256 contractBalance = address(this).balance;
    (bool success, ) = owner().call{value: contractBalance}("");
    require(success, "Emergency withdraw failed");
}
```

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

This issue has been acknowledged by the team.

2. Incorrect stakeIndex emitted in Staked event

Severity: Low	Category: Logging
Target: - contracts/StakingPool.sol	

Description

The stake function stores the new position under the current `_globalStakeIndex`, but emits the Staked event with the incremented value of `_globalStakeIndex`. This introduces an off-by-one error between the on-chain storage index and the event log.

contracts/StakingPool.sol:L74-L100

```
function stake(uint256 tierIndex) external payable {
    …
    _userStakes[_msgSender()].push(_globalStakeIndex);

    _globalStakeIndex++;

    emit Staked(
        _msgSender(),
        _globalStakeIndex,
        msg.value,
        tier.duration,
        tier.multiplier
    );
}
```

Recommendation

Emit the event first, then increment `_globalStakeIndex` to ensure ID consistency.

Status

This issue has been acknowledged by the team.

3. SupportsInterface breaks ERC165 compatibility	
Severity: Low	Category: Configuration
Target: - contracts/StakingPool.sol	

Description

The contract inherits ERC165 but overrides supportsInterface without delegating to the parent implementation:

contracts/StakingPool.sol:L64-L68

```
function supportsInterface(
   bytes4 interfaceId
) public view virtual override returns (bool) {
   return interfaceId == type(IStakingPool).interfaceId;
}
```

As a result, the contract returns false for `type(IERC165).interfaceId` and any other interfaces declared by parent contracts, even though it is structurally an ERC165 contract. This violates the ERC165 expectation that a compliant contract reports support for the ERC165 interface ID, and may cause other contracts or tools that rely on ERC165 detection to misclassify this contract and disable integrations.

Recommendation

Update supportsInterface to combine the custom interface with the parent ERC165 logic.

Status

This issue has been acknowledged by the team.

2.3 Informational Findings

4. Comment mismatch code in multiplier usage	
Severity: Informational	Category: Inconsistency
Target: - contracts/StakingPool.sol - contracts/IStakingPool.sol	

Description

There is an inconsistency between the code and comment:

contracts/IStakingPool.sol:L12-L15

```
struct StakingTier {
   uint256 duration; // Duration in seconds
   uint256 multiplier; // Multiplier scaled by 1000 (e.g., 1300 = 1.3x)
}
```

contracts/StakingPool.sol:L89-L108

```
function claim(uint256 stakeIndex) external nonReentrant {
   ...
   //  duration * principal * (apr / 365days)
   //  = duration * principal * ( stakeInfo.stakingTier.multiplier / (100 *
BASE_MULTIPLIER) / 365days)
   uint256 rewards = (stakeInfo.stakingTier.duration * stakeInfo.principal *
stakeInfo.stakingTier.multiplier)
       / (365 days) / (100 * BASE_MULTIPLIER);
   ...
}
```

Recommendation

Update the comment to be consistent.

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files:

File	SHA-1 hash
StakingPool.sol	7f105ea490926c09c32476df7acee8c6135b8095
dep.sol	a2ef65e3b6126a3fe9b287c151bb8c3178c94ab4ps
IStakingPool.sol	74faeb51960dc30c7f7d7c68b39d3d27a1982251