PALADIN
BLOCKCHAIN SECURITY

# Smart Contract Security Assessment

Preliminary Report

## For DragonSwap (Competition)

27 January 2025

paladinsec.co

info@paladinsec.co

# Table of Contents

# Disclaimer

Paladin Blockchain Security ("Paladin") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Paladin.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Paladin is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Paladin or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Paladin retains the right to re-use any and all knowledge and expertise gained during the audit process, including, but not limited to, vulnerabilities, bugs, or new attack vectors. Paladin is therefore allowed and expected to use this knowledge in subsequent audits and to inform any third party, who may or may not be our past or current clients, whose projects have similar vulnerabilities. Paladin is furthermore allowed to claim bug bounties from third-parties while doing so.

# 1 Overview

This report has been prepared for DragonSwap's Competition contracts on the SEI network. It is a diff-audit of the changes in the contract compared with a previous audit done by Paladin. Paladin provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

## 1.1 Summary

| | |
|---|---|
| **Project Name** | DragonSwap |
| **URL** | https://dragonswap.app |
| **Platform** | SEI |
| **Language** | Solidity |
| **Preliminary Contracts** | https://github.com/dragonswap-app/comp-contracts/commit/c26bb10a95c38d1ffafcf2f7bd599db02c6953ec |
| **Resolution** | https://github.com/dragonswap-app/comp-contracts/commit/8ffbfc8c17534cb73b21b12937dda3f3b6887ac5 |

## 1.2 Contracts Assessed

| Name | Contract | Live Code Match |
|---|---|---|
| `Competition` | | |
| `Factory` | | |

## 1.3 Findings Summary

| Severity | Found | Resolved | Partially Resolved | Acknowledged (no change made) |
|---|---|---|---|---|
| Governance | 0 | - | - | - |
| High | 0 | - | - | - |
| Medium | 0 | - | - | - |
| Low | 0 | - | - | - |
| Informational | 5 | 5 | - | - |
| **Total** | **5** | **5** | - | **0** |

### Classification of Issues

| Severity | Description |
|---|---|
| Governance | Issues under this category are where the governance or owners of the protocol have certain privileges that users need to be aware of, some of which can result in the loss of user funds if the governance's private keys are lost or if they turn malicious, for example. |
| High | Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency. |
| Medium | Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible. |
| Low | Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless. |
| Informational | Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any. |

### 1.3.1 Competition

| ID | Severity | Summary | Status |
|---|---|---|---|
| 01 | INFO | Unused `MINIMAL_DEPOSIT` constant after deposit validation update | RESOLVED |
| 02 | INFO | Gas optimizations | RESOLVED |
| 03 | INFO | `addSwapTokens` does not revert if the swap token has already been added | RESOLVED |
| 04 | INFO | Insufficient validation | RESOLVED |

### 1.3.2 Factory

| ID | Severity | Summary | Status |
|---|---|---|---|
| 05 | INFO | Gas optimizations | RESOLVED |

# 2 Findings

## 2.1 Competition

`Competition` is a trading competition platform where users can participate in token swapping activities within a defined time window (between `startTimestamp` and `endTimestamp`). The team is able to extend the `endTimestamp` as long as the current `endTimestamp` has not been reached.

Users first deposit approved stablecoins to enter the competition, after which they can perform various types of token swaps (V1 and V2) using supported swap tokens. The contract integrates with a router for executing swaps, tracks user balances, and allows participants to exit the competition by withdrawing their tokens.

The contract is designed to be deployed by a factory contract and can be managed by an owner who can add new swap tokens.

### 2.1.1 Privileged Functions

- `initialize`
- `addSwapTokens`

## 2.1.2 Issues & Recommendations

| Issue #01 | Unused `MINIMAL_DEPOSIT` constant after deposit validation update |
|---|---|
| **Location** | https://github.com/dragonswap-app/comp-contracts/blob/4f618ad9af0f5370da28f31b57fdb1408a5c5092/src/Competition.sol#L48<br>`uint256 public constant MINIMAL_DEPOSIT = 10e6;` |
| **Severity** | INFORMATIONAL |
| **Description** | The contract defines a constant `MINIMAL_DEPOSIT = 10e6` that is no longer used after the deposit validation logic was updated to use token decimals. This creates confusion and potential maintenance issues as the constant remains in the code but has been superseded by a different implementation. The constant is defined but never used, as the deposit validation now correctly uses the token's decimals to determine the minimum deposit amount. |
| **Recommendation** | Remove the unused constant to improve code clarity and prevent confusion. |
| **Resolution** | RESOLVED |

| Issue #02 | Gas optimizations |
|---|---|
| **Severity** | INFORMATIONAL |
| **Description** | `L73-74, L324`<br><br>The contract's token addition functions (`addSwapTokens` and `initialize`) currently use memory for array parameters when `calldata` would be more efficient. Since these arrays are only used for reading values and not modified within the functions, using memory creates unnecessary gas costs by copying the arrays to memory.<br><br>Replace memory with `calldata` for array parameters in external functions and their internal helpers since the arrays are only used for reading.<br><br>—<br>L295<br><br>Using named return variables in Solidity functions instead of explicitly declaring return variables can save approximately 9 gas per variable. This optimization works by pre-allocating memory for return values and avoiding additional stack operations.<br><br>Reference: https://x.com/DevDacian/status/1796396988659093968<br><br>Consider changing the code to:<br>`function isSwapToken(address token) public view returns (bool isToken) {`<br>`  isToken = swapTokenIds[token] > 0;`<br>`}` |
| **Recommendation** | Consider implementing the above recommendations. |
| **Resolution** | RESOLVED |

| Issue #03 | **addSwapTokens does not revert if the swap token has already been added** |
|---|---|
| **Severity** | INFORMATIONAL |
| **Description** | As `addSwapTokens` does not revert if the swap token has already been added, an existing swap token can be mistakenly added again with a true for `_stableCoins`, making it change from false to true in `stablecoins`. |

```
function _addSwapTokens(address[] memory _swapTokens, bool
_stableCoins) private {
    // Gas opt
    uint256 _length = _swapTokens.length;
    uint256 length = swapTokens.length;
    for (uint256 i; i < _length; ++i) {
        address _token = _swapTokens[i];
        // Ensure there is code at the specified address
        Utils._isContract(_token);
        if (_stableCoins) {
            stableCoins[_token] = true;
            emit StableCoinAdded(_token);
        }
        // Add token if it is not already present
        if (!isSwapToken(_token)) {
            swapTokenIds[_token] = length++;
            swapTokens.push(_token);
            emit SwapTokenAdded(_token);
        }
    }
}
```

This can be prevented if it reverts when `isSwapToken(_token)` is true, preventing the stablecoin status of an existing token to be changed.

| | |
|---|---|
| **Recommendation** | Consider reverting when `isSwapToken(_token)` is true. |
| **Resolution** | RESOLVED |

| Issue #04 | Insufficient validation |
|---|---|
| **Severity** | INFORMATIONAL |
| **Description** | For all the swap related functions, check that in and out token are not the same address inside `_validateSwapAndApprove`. |
| **Recommendation** | Consider implementing the recommended validation. |
| **Resolution** | RESOLVED |

## 2.2 Factory

`Factory` is a deployment manager for competition contracts, implementing a clone factory pattern. It allows an owner to deploy new instances of a competition contract using a minimal proxy pattern (EIP-1167), which creates lightweight clones of a master implementation contract.

The factory keeps track of all deployments, maintains a reference to the current implementation, and provides functions to query deployment information.

### 2.2.1 Privileged Functions

- `setImplementation`
- `deploy`

## 2.2.2 Issues & Recommendations

| Issue #05 | Gas optimization |
|---|---|
| **Severity** | INFORMATIONAL |
| **Description** | Functions without named return variables require additional stack operations, consuming unnecessary gas. This inefficiency compounds with frequent function calls.<br><br>Consider changing this:<br>`function noOfDeployments() public view returns (uint256) {`<br>`  [...]`<br><br>to this:<br>`function noOfDeployments() public view returns (uint256 count) {`<br>`  count = deployments.length;`<br>`}`<br><br>Additionally, consider changing this:<br>`function getLatestDeployment() external view returns (address) {`<br>`  [...]`<br><br>to this:<br>`function getLatestDeployment() external view returns (address latestDeployment) {`<br>`  uint256 _noOfDeployments = noOfDeployments();`<br>`  if (_noOfDeployments > 0) latestDeployment = deployments[_noOfDeployments - 1];`<br>`}`<br><br>`  uint256 index = 0;`<br>`  for (uint256 i = startIndex; i <= endIndex; i++) {`<br>`  _deployments[index] = deployments[i];`<br>`  index++;`<br>`  }`<br>`}` |
| **Recommendation** | Consider making the recommended changes. |
| **Resolution** | RESOLVED |

PALADIN
BLOCKCHAIN SECURITY