L3X Pre Launch Audit Report

Apr 22, 2024

Table of Contents

Summary

This report has been prepared for L3X Pre Launch smart contract, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Overview

Project Summary

Project Name	**L3X Pre Launch**
Codebase	**https://github.com/L3X-Protocol/l3x-pre-launch-contracts**
Commit	**b0c353b63a39a5368e09900760dbb87fba46c7ee**
Language	**Solidity**

Audit Summary

Delivery Date	**Apr 22, 2024**
Audit Methodology	**Static Analysis, Manual Review**
Total Isssues	**5**

[WP-L1] `getUserStakedBalances()` should include all user stakes even if a token is removed from `acceptedTokens` .

Low

Issue Description

https://github.com/L3X-Protocol/l3x-pre-launch-contracts/blob/3b00fbb9f46abdb10c99e9b82b17cbf7e2b5d0db/contracts/PreLaunchStaking.sol#L167-L182

```
167  function getUserStakedBalances(address _user) external view returns (address[]
     memory, uint256[] memory) {
168      uint256 count = acceptedTokenCount();
169      address[] memory stakedTokens = new address[](count);
170      uint256[] memory stakedBalances = new uint256[](count);
171
172      uint256 index = 0;
173      for (uint256 i = 0; i < acceptedTokensArray.length; ++i) {
174          address acceptedToken = acceptedTokensArray[i];
175          if (acceptedTokens[acceptedToken]) {
176              stakedTokens[index] = acceptedToken;
177              stakedBalances[index] = userStakes[_user][acceptedToken];
178              ++index;
179          }
180      }
181      return (stakedTokens, stakedBalances);
182  }
```

Recommendation

Consider removing L175.

Status

✓ Fixed

[WP-L2] `getAllAcceptedTokens` , `acceptedTokenCount` , and `getUserStakedBalances` may return duplicate results.

Low

Issue Description

`removeToken` does not actually remove elements from `acceptedTokensArray` . If a `token` is added again after being `removed` , it causes duplicates in `acceptedTokensArray` , resulting in `getAllAcceptedTokens` , `acceptedTokenCount` , and `getUserStakedBalances` returning duplicate results.

https://github.com/L3X-Protocol/l3x-pre-launch-contracts/blob/3b00fbb9f46abdb10c99e9b82b17cbf7e2b5d0db/contracts/PreLaunchStaking.sol#L222-L230

```
222     /**
223      * @dev Owner can remove a token from being acceptable for staking.
224      * @param _token Address of the token to be removed.
225      */
226     function removeToken(address _token) external onlyOwner {
227         require(acceptedTokens[_token], "removeToken: token not whitelisted");
228         acceptedTokens[_token] = false;
229         emit TokenRemoved(_token);
230     }
```

https://github.com/L3X-Protocol/l3x-pre-launch-contracts/blob/3b00fbb9f46abdb10c99e9b82b17cbf7e2b5d0db/contracts/PreLaunchStaking.sol#L132-L182

```
132     /**
133      * @notice A helper function to get all accepted tokens
134      * @return Array of addresses representing accepted tokens
135      */
136     function getAllAcceptedTokens() external view returns (address[] memory) {
137         address[] memory result = new address[](acceptedTokenCount());
138         uint256 index = 0;
139         for (uint256 i = 0; i < acceptedTokensArray.length; i++) {
140             if (acceptedTokens[acceptedTokensArray[i]]) {
141                 result[index] = acceptedTokensArray[i];
```

```
142                 index++;
143             }
144         }
145         return result;
146     }
147 
148     /**
149      * @notice A helper function to get the count of accepted tokens
150      * @return Number of accepted tokens
151      */
152     function acceptedTokenCount() public view returns (uint256) {
153         uint256 count = 0;
154         for (uint256 i = 0; i < acceptedTokensArray.length; ++i) {
155             if (acceptedTokens[acceptedTokensArray[i]]) {
156                 ++count;
157             }
158         }
159         return count;
160     }
161 
162     /**
163      * @notice A helper function to get all of a user's all staked balances
164      * @param _user Address of the user
165      * @return Arrays of addresses and uint256 representing staked tokens and
    their balances
166      */
167     function getUserStakedBalances(address _user) external view returns (address[]
    memory, uint256[] memory) {
168         uint256 count = acceptedTokenCount();
169         address[] memory stakedTokens = new address[](count);
170         uint256[] memory stakedBalances = new uint256[](count);
171 
172         uint256 index = 0;
173         for (uint256 i = 0; i < acceptedTokensArray.length; ++i) {
174             address acceptedToken = acceptedTokensArray[i];
175             if (acceptedTokens[acceptedToken]) {
176                 stakedTokens[index] = acceptedToken;
177                 stakedBalances[index] = userStakes[_user][acceptedToken];
178                 ++index;
179             }
180         }
181         return (stakedTokens, stakedBalances);
182     }
```

Recommendation

Consider changing `acceptedTokensArray` from Array to EnumerableSet.AddressSet to avoid duplicates.

Status

✓ Fixed

[WP-G3] Caching storage reads can save gas

Gas

Issue Description

By reusing the storage read result from L140 in L141, we can save one hot storage read per loop iteration.

https://github.com/L3X-Protocol/l3x-pre-launch-contracts/blob/3b00fbb9f46abdb10c99e9b82b17cbf7e2b5d0db/contracts/PreLaunchStaking.sol#L36

```
36      address[] private acceptedTokensArray;
```

https://github.com/L3X-Protocol/l3x-pre-launch-contracts/blob/3b00fbb9f46abdb10c99e9b82b17cbf7e2b5d0db/contracts/PreLaunchStaking.sol#L132-L146

```
132     /**
133      * @notice A helper function to get all accepted tokens
134      * @return Array of addresses representing accepted tokens
135      */
136     function getAllAcceptedTokens() external view returns (address[] memory) {
137         address[] memory result = new address[](acceptedTokenCount());
138         uint256 index = 0;
139         for (uint256 i = 0; i < acceptedTokensArray.length; i++) {
140             if (acceptedTokens[acceptedTokensArray[i]]) {
141                 result[index] = acceptedTokensArray[i];
142                 index++;
143             }
144         }
145         return result;
146     }
```

Status

✓ Fixed

[WP-N4] Consider `_disableInitializers()` in `PreLaunchStaking#constructor()`

Issue Description

It is a best practice to call `_disableInitializers()` in the constructor function of an upgradeable contract.

See: https://docs.openzeppelin.com/upgrades-plugins/1.x/writing-upgradeable#initializing_the_implementation_contract

Recommendation

```
/// @custom:oz-upgrades-unsafe-allow constructor
constructor() {
    _disableInitializers();
}
```

Status

ⓘ Acknowledged

[WP-N5] Consider using `SafeERC20.forceApprove()` instead of `IERC20.approve()` .

Issue Description

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v5.0.2/contracts/token/ERC20/utils/SafeERC20.sol#L73-L74

```
71      /**
72       * @dev Set the calling contract's allowance toward `spender` to `value`. If `token` returns no value,
73       * non-reverting calls are assumed to be successful. Meant to be used with tokens that require the approval
74       * to be set to zero before setting it to a non-zero value, such as USDT.
75       */
76      function forceApprove(IERC20 token, address spender, uint256 value) internal {
77          bytes memory approvalCall = abi.encodeCall(token.approve, (spender, value));
78
79          if (!_callOptionalReturnBool(token, approvalCall)) {
80              _callOptionalReturn(token, abi.encodeCall(token.approve, (spender, 0)));
81              _callOptionalReturn(token, approvalCall);
82          }
83      }
```

https://github.com/L3X-Protocol/l3x-pre-launch-contracts/blob/3b00fbb9f46abdb10c99e9b82b17cbf7e2b5d0db/contracts/PreLaunchStaking.sol#L105-L126

```
105     /**
106      * @notice Allow users to bridge their assets to L3 after the bridge is established
107      * @param _token Address of the token to bridge
108      * @param _minGasLimit Minimum gas limit for each individual withdrawal transaction
109      * @param _receiver The receiver of the funds on L3
110      */
111     function bridgeAsset(address _token, uint32 _minGasLimit, address _receiver) external whenNotPaused nonReentrant {
```

```
112         address bridgeAddress = bridgeProxyAddress;
113         require(bridgeAddress != address(0), "Bridge not ready");
114         uint256 transferAmount = userStakes[msg.sender][_token];
115         require(transferAmount != 0, "Withdrawal completed or token never
    staked");
116         require(acceptedTokens[_token], "token not accepted");
117
118         userStakes[msg.sender][_token] = 0;
119         stakedAmounts[_token] -= transferAmount;
120
121         // bridge ERC20 token
122         IERC20(_token).approve(bridgeAddress, transferAmount);
123         BridgeInterface(bridgeAddress).depositERC20To(_token, _receiver,
    transferAmount, _minGasLimit, hex"");
124
125         emit AssetBridged(msg.sender, _token, _receiver, transferAmount);
126     }
```

Status

✓ Fixed

Appendix

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by WatchPug; however, WatchPug does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Smart Contract technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.