METATRUST

Security Assessment for

4everland

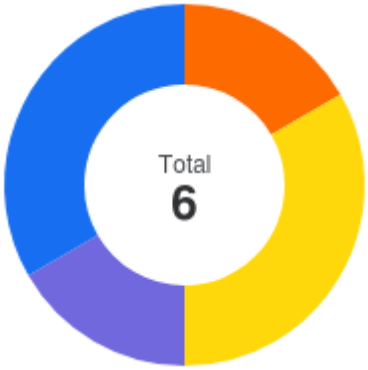
November 01, 2024

Executive Summary

Overview	
Project Name	4everland
Codebase URL	https://github.com/4everland/land/tree/v3
Scan Engine	Security Analyzer
Scan Time	2024/11/01 08:00:00
Commit Id	e900538ce91c39173cff68e2aa5a57715a62fde1

Total	
Critical Issues	0
High risk Issues	1
Medium risk Issues	2
Low risk Issues	1
Informational Issues	2

Critical Issues	The issue can cause large economic losses, large-scale data disorder, loss of control of authority management, failure of key functions, or indirectly affect the correct operation of other smart contracts interacting with it.
High Risk Issues	The issue puts a large number of users' sensitive information at risk or is reasonably likely to lead to catastrophic impacts on clients' reputations or serious financial implications for clients and users.
Medium Risk Issues	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 moderate financial impact.
Low Risk Issues	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 Issue	The issue does not pose an immediate risk but is relevant to security best practices or Defence in Depth.



Critical Issues	0%	0
High risk Issues	17%	1
Medium risk Issues	33%	2
Low risk Issues	17%	1
Informational Issues	33%	2

Summary of Findings

MetaScan security assessment was performed on **November 01, 2024 08:00:00** on project **4everland** with the repository on branch **default branch**. The assessment was carried out by scanning the project's codebase using the scan engine **Security Analyzer**. There are in total **6** vulnerabilities / security risks discovered during the scanning session, among which **1** high risk vulnerabilities, **2** medium risk vulnerabilities, **1** low risk vulnerabilities, **2** informational issues.

ID	Description	Severity	Alleviation
MSA-001	Missing Invoking The `configureClaimableGas` For The Blast Contract	High risk	Fixed
MSA-002	Centralization Risk	Medium risk	Acknowledged
MSA-003	Different Coins Have The Same Weight When Calculating The `landAmount`	Medium risk	Mitigated
MSA-004	The `getPriceUnsafe()` Returned Price Update May be Arbitrarily Far in The Past.	Low risk	Acknowledged
MSA-005	Unused event	Informational	Acknowledged
MSA-006	Missing Zero Address Check	Informational	Acknowledged

Findings

High risk (1)

1. Missing Invoking The `configureClaimableGas` For The Blast Contract

High risk Security Analyzer

The `claimAllGas()` function tries to let the owner to claim gas. But, the `configureClaimableGas()` function that sets the Gas Mode for the contract to claimable does not be executed.

The `BLAST.configureClaimableGas()` should be invoked first to claim the gas.

Document: https://docs.blast.io/building/guides/gas-fees

File(s) Affected

contracts/oracleland/BlastOracleLand.sol #46-48

```
46      function claimAllGas(address to) external onlyOwner {
47          blast.claimAllGas(address(this), to);
48      }
```

Recommendation

Invoking the `BLAST.configureClaimableGas()` function in the constructor.

Alleviation Fixed

The team fixed this finding, in the commit c8b5b586a30b79e5084b942ff6defd10fb381c1a.

Medium risk (2)

1. Centralization Risk

Medium risk Security Analyzer

In the `Land` contract, the owner has the privilege of the following functions:

- `withdraw`: Allows the owner to withdraw tokens from the contract.

In the `Land` contract, the guardian has the privilege of the following functions:

- `addCoin`: Allows the owner to add a new coin to the contract.
- `removeCoin`: Allows the owner to remove a coin from the contract.

In the `LandCore` contract, the owner has the privilege of the following functions:

- `setGuardian`: Set the guardian address for the contract;
- `transferOwnership`: Transfer ownership of the contract to a new address.

In the `BlastOracleLand` contract, the owner has the privilege of the following functions:

- `claimAllETHYield`: Claim all ETH yield from the Blast protocol;
- `claimAllUSDBYield`: Claim all USDB yield from the Blast protocol;
- `claimAllGas`: Claim all gas from the Blast protocol.

In the `OracleLand` contract, the owner has the privilege of the following functions:

- `setPriceFeed`: Sets the price feed contract to fetch prices from.

In the `ChainlinkPriceFeed` contract, the owner has the privilege of the following functions:

- `setOracle`: Allows the owner to set the oracle for a token along with heartbeat value.

In the `FixedPriceFeed` contract, the owner has the privilege of the following functions:

- `setPrice`: Set the price for a specific token;
- `fetchPrice`: Fetch the price of a specific token.

In the `PythPriceFeed` contract, the owner has the privilege of the following functions:

- `setOracle`: Allows the owner to set the oracle for a token with a specific Pyth contract, feed, and heartbeat value.

File(s) Affected

contracts/core/Land.sol #9-9

```
9  contract Land is ILand, LandOwnableUpgradeable {
```

contracts/core/LandCore.sol #7-7

```
7  contract LandCore is ILandCore {
```

contracts/oracleland/BlastOracleLand.sol #8-8

```
8  contract BlastOracleLand is OracleLand {
```

contracts/oracleland/OracleLand.sol #9-9

```
9  contract OracleLand is Land {
```

contracts/pricefeeds/ChainlinkPriceFeed.sol #11-11

```
11  contract ChainlinkPriceFeed is LandOwnable {
```

contracts/pricefeeds/FixedPriceFeed.sol #11-11

```
11  contract FixedPriceFeed is LandOwnable {
```

contracts/pricefeeds/PythPriceFeed.sol #10-10

```
10  contract PythPriceFeed is LandOwnable {
```

Recommendation

Consider implementing a decentralized governance mechanism or a multi-signature scheme that requires consensus among multiple parties before pausing or unpausing the contract. This can help mitigate the centralization risk associated with a single owner controlling critical contract functions. Alternatively, you can provide a clear justification for the centralization aspect and ensure that users are aware of the potential risks associated with a single point of control.

Alleviation Acknowledged

The team acknowledged this finding.

2. Different Coins Have The Same Weight When Calculating The `landAmount`

Medium risk Security Analyzer

The `mint` function allows users to accumulating the land amount by costing kinds of coins. The point is that, the calculating of `landAmount` do not take the coin price into account:

```
    function mint(ICoin coin, bytes32 account, uint256 amount) external whenNotPaused {
        ...
        uint256 coinAmount = formatValue(coin, amount);
```

```
        uint256 landAmount = coinAmount * landPerCoin;
        balances[account] += landAmount;
```

It implies that all the coins have the same weight/price.

File(s) Affected

contracts/core/Land.sol #23-30

```
23      function mint(ICoin coin, bytes32 account, uint256 amount) external whenNotPaused {
24          require(coinExists(coin), "Land: nonexistent coin");
25          (bool success1,) = address(coin).call(abi.encodeWithSignature("transferFrom(address,address,uint
26          require(success1, "Land: transfer from failed");
27          uint256 coinAmount = formatValue(coin, amount);
28          uint256 landAmount = coinAmount * landPerCoin;
29          balances[account] += landAmount;
30          deposits[account][coin] += amount;
```

Recommendation

Checking if it is an intended design.

Alleviation Mitigated

The team replied that they use different smart contracts to deal with stable coins and the native token, the contract `core/Land.sol` deals with the stable coin, and the contract `oracleland/OracleLand` deals with the native token.

Low risk (1)

1. The `getPriceUnsafe()` Returned Price Update May be Arbitrarily Far in The Past.

Low risk Security Analyzer

The `getPriceUnsafe` function is unsafe as the returned price update may be arbitrarily far in the past.

```
interface IPyth is IPythEvents {
    /// @notice Returns the price of a price feed without any sanity checks.
    /// @dev This function returns the most recent price update in this contract without any recency checks.
    /// This function is unsafe as the returned price update may be arbitrarily far in the past.
    ///
    /// Users of this function should check the `publishTime` in the price to ensure that the returned price is
    /// sufficiently recent for their application. If you are considering using this function, it may be
    /// safer / easier to use `getPriceNoOlderThan`.
    /// @return price - please read the documentation of PythStructs.Price to understand how to use this safely.
    function getPriceUnsafe(
        bytes32 id
    ) external view returns (PythStructs.Price memory price);
```

Reference: https://github.com/pyth-network/pyth-crosschain/blob/b0aa4b10317b609f72bb5757f6e770cee243f585/target_chains/ethereum/sdk/solidity/IPyth.sol#L10-L21

File(s) Affected

contracts/pricefeeds/PythPriceFeed.sol #136-146

```
136     function _fetchCurrentFeedResponse(IPyth _priceAggregator, bytes32 _feed) internal view returns (Fee
137         try _priceAggregator.getPriceUnsafe(_feed) returns (IPyth.Price memory price) {
138             response.price = price.price;
139             response.conf = price.conf;
140             response.expo = price.expo;
141             response.publishTime = price.publishTime;
142             response.success = true;
143         } catch {
144             return response;
145         }
146     }
```

Recommendation

Consider accessing real-time asset data using Pyth Price Feeds.

```
// Update the prices to the latest available values and pay the required fee for it. The `priceUpdateData` data
// should be retrieved from our off-chain Price Service API using the `pyth-evm-js` package.
// See section "How Pyth Works on EVM Chains" below for more information.
uint fee = pyth.getUpdateFee(priceUpdateData);
 pyth.updatePriceFeeds{ value: fee }(priceUpdateData);
```

Reference: https://docs.base.org/tutorials/oracles-pyth-price-feeds/

Alleviation Acknowledged

The team acknowledged this finding.

Informational (2)

1. Unused event

Informational Security Analyzer

The presence of event that is declared but never used in the codebase. It may increase computation costs and lead to unnecessary gas consumption.

File(s) Affected

contracts/pricefeeds/PythPriceFeed.sol #44-44

```
44      event PriceFeedStatusUpdated(address token, address oracle, bool isWorking);
```

Recommendation

Remove unused event or emit them in the right place to avoid negative effects and improve code readability if there is no plan for further usage.

Alleviation Acknowledged

The team acknowledged this finding.

2. Missing Zero Address Check

Informational Security Analyzer

Contracts, like LandCore, missing zero address check for key state variables. Example:

```
    //LandCore.sol
    constructor(address _owner, address _guardian) {
```

```
        owner = _owner;
        guardian = _guardian;
        emit GuardianSet(_guardian);
    }
```

Key addresses assignment are recommended to adding the zero address check to prevent potential risk.

File(s) Affected

contracts/core/LandCore.sol #14-18

```
14      constructor(address _owner, address _guardian) {
15          owner = _owner;
16          guardian = _guardian;
17          emit GuardianSet(_guardian);
18      }
```

Recommendation

Adding zero address check on all the contracts for the key state variables.

Alleviation Acknowledged

The team acknowledged this finding.

Audit Scope

File	SHA256	File Path
OracleLand.sol	c8fe9605a702fda60bccbebb2be36e2457772123ded9f33f17a12b7c336f214d	/contracts/oracleland/OracleLand.sol
BlastOracleLand.sol	a8eabfee2dd354cf046b3b1e63507c8a804a17caaae7224ab9cf9bd97a0bb9c9	/contracts/oracleland/BlastOracleLand.sol
Land.sol	6e0cfe546eb3eb219f8d22cf3d65bd7cad39ef48d9e336cff54c1b4ce6aa34d0	/contracts/core/Land.sol
LandCore.sol	b5fdac233ad7bcfb49ee877c42497cce99029a0c2fe7b000a0c12f778149637c	/contracts/core/LandCore.sol
PythPriceFeed.sol	12338fc4b2d5902d5852e9d687eda061c70ae51b6fcd94de2f2389232d0dd431	/contracts/pricefeeds/PythPriceFeed.sol
FixedPriceFeed.sol	83d37f05a69b71f5b7053a1be3d7d9777bf3edd3ec7403ab8598bf506356b0e0	/contracts/pricefeeds/FixedPriceFeed.sol
ChainlinkPriceFeed.sol	a393269c2dcd2a20bef85a1bc73e505c96a0a11dcdd21fce11f22d6ce23ff5a2	/contracts/pricefeeds/ChainlinkPriceFeed.sol
LandOwnable.sol	25c6fa48ea599b8c984a56c2ba1f1f5aa621a43fe583d778acb7162b1e4f2cba	/contracts/dependencies/LandOwnable.sol
console.sol	c9fa628da7d3b789a019a071b0e32e477e320daf80466aefea390f7c16dfad81	/contracts/dependencies/console.sol
LandOwnableUpgradeable.sol	2e7c729ca175f1bb7e312f7dc95c2214a5ed871dd6dbf32eb3a3e4e235c4e507	/contracts/dependencies/LandOwnableUpgradeable.sol

Disclaimer

This report is governed by the stipulations (including but not limited to service descriptions, confidentiality, disclaimers, and liability limitations) outlined in the Services Agreement, or as detailed in the scope of services and terms provided to you, the Customer or Company, within the context of the Agreement. The Company is permitted to use this report only as allowed under the terms of the Agreement. Without explicit written permission from MetaTrust, this report must not be shared, disclosed, referenced, or depended upon by any third parties, nor should copies be distributed to anyone other than the Company.

It is important to clarify that this report neither endorses nor disapproves any specific project or team. It should not be viewed as a reflection of the economic value or potential of any product or asset developed by teams or projects engaging MetaTrust for security evaluations. This report does not guarantee that the technology assessed is completely free of bugs, nor does it comment on the business practices, models, or legal compliance of the technology's creators.

This report is not intended to serve as investment advice or a tool for investment decisions related to any project. It represents a thorough assessment process aimed at enhancing code quality and mitigating risks inherent in cryptographic tokens and blockchain technology. Blockchain and cryptographic assets inherently carry ongoing risks. MetaTrust's role is to support companies and individuals in their security diligence and to reduce risks associated with the use of emerging and evolving technologies. However, MetaTrust does not guarantee the security or functionality of the technologies it evaluates.

MetaTrust's assessment services are contingent on various dependencies and are continuously evolving. Accessing or using these services, including reports and materials, is at your own risk, on an as-is and as-available basis. Cryptographic tokens are novel technologies with inherent technical risks and uncertainties. The assessment reports may contain inaccuracies, such as false positives or negatives, and unpredictable outcomes. The services may rely on multiple third-party layers.

All services, labels, assessment reports, work products, and other materials, or any results from their use, are provided "as is" and "as available," with all faults and defects, without any warranty. MetaTrust expressly disclaims all warranties, whether express, implied, statutory, or otherwise, including but not limited to warranties of merchantability, fitness for a particular purpose, title, non-infringement, and any warranties arising from course of dealing, usage, or trade practice. MetaTrust does not guarantee that the services, reports, or materials will meet specific requirements, be error-free, or be compatible with other software, systems, or services.

Neither MetaTrust nor its agents make any representations or warranties regarding the accuracy, reliability, or currency of any content provided through the services. MetaTrust is not liable for any content inaccuracies, personal injuries, property damages, or any loss resulting from the use of the services, reports, or materials.

Third-party materials are provided "as is," and any warranty concerning them is strictly between the Customer and the third-party owner or distributor. The services, reports, and materials are intended solely for the Customer and should not be relied upon by others or shared without MetaTrust's consent. No third party or representative thereof shall have any rights or claims against MetaTrust regarding these services, reports, or materials.

The provisions and warranties of MetaTrust in this agreement are exclusively for the Customer's benefit. No third party has any rights or claims against MetaTrust regarding these provisions or warranties. For clarity, the services, including any assessment reports or materials, should not be used as financial, tax, legal, regulatory, or other forms of advice.