Competitive Security Assessment

Loxodrome

Apr 11th, 2024

Secure3

secure3.io

Summary

This report is prepared for the project to identify vulnerabilities and issues in the smart contract source code. A group of NDA covered experienced security experts have participated in the Secure3's Audit Contest to find vulnerabilities and optimizations. Secure3 team has participated in the contest process as well to provide extra auditing coverage and scrutiny of the finding submissions.

The comprehensive examination and auditing scope includes:

• Cross checking contract implementation against functionalities described in the documents and white paper disclosed by the project owner.

• Contract Privilege Role Review to provide more clarity on smart contract roles and privilege.

• Using static analysis tools to analyze smart contracts against common known vulnerabilities patterns.

• Verify the code base is compliant with the most up-to-date industry standards and security best practices.

• Comprehensive line-by-line manual code review of the entire codebase by industry experts.

The security assessment resulted in findings that are categorized in four severity levels: Critical, Medium, Low, Informational. For each of the findings, the report has included recommendations of fix or mitigation for security and best practices.

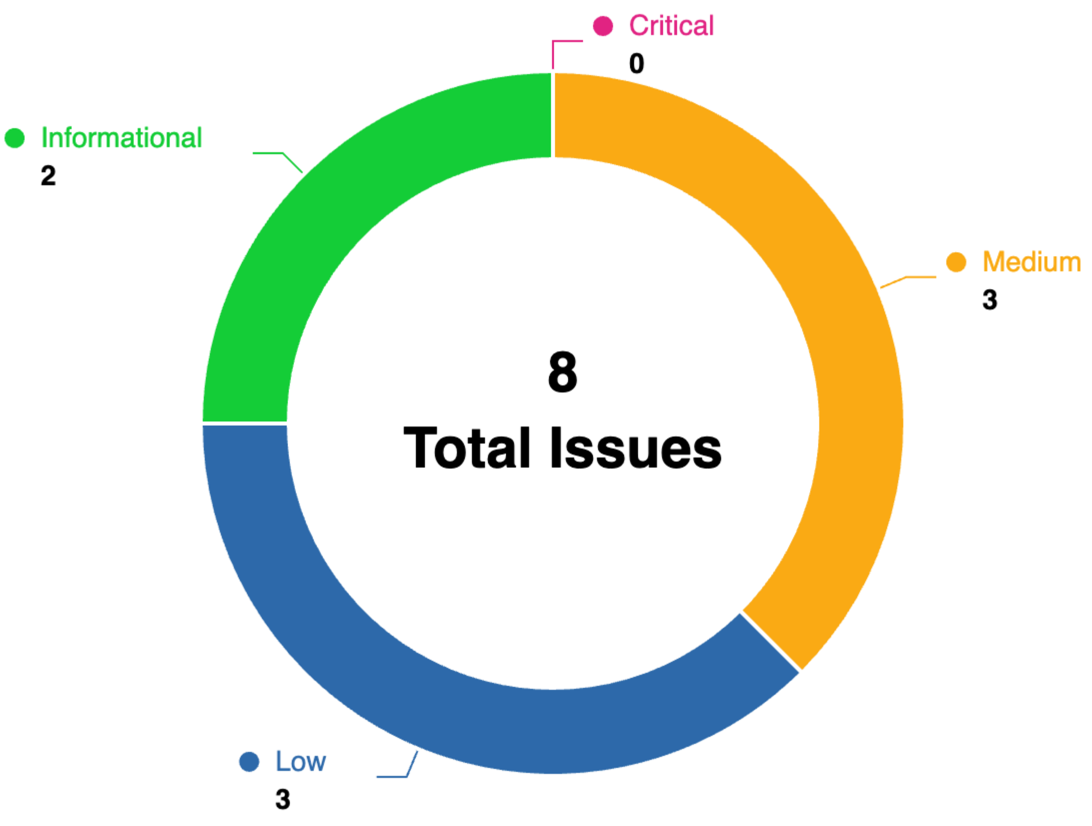
Overview

Project Name	Loxodrome
Language	Solidity
Codebase	• https://github.com/Loxodromexyz/Loxodrome_Contracts/ • audit version - 8f65ccb6a6b6c8f7668b697f4b089e1054e08509 • final version - 1b38bcdc0696a7e3512944c424fa45957d065304
Audit Methodology	• Audit Contest • Business Logic and Code Review • Privileged Roles Review • Static Analysis

Audit Scope

File	SHA256 Hash
./Loxodrome_Contracts/contracts/MasterChef.sol	04684860156c93cd9ade75587af53035fb0e720d84b4551e62ec00b69b51cd2f
./Loxodrome_Contracts/contracts/Minter.sol	69ec41896f54e5e4e69f57624439c679e7b78eb7641263055fadf71c959c0cc4
./Loxodrome_Contracts/contracts/RewardsDistributor.sol	782abfb60d53e526435a0fbbb8b709c2e38e7374ea5dd9a2b638b9ca2baf2560
./Loxodrome_Contracts/contracts/VoterV2.sol	5a77be2afaf453911a32069ef6f84a87e9c7525b322b07ec9f8327755201d9cf

Code Assessment Findings



ID	Name	Category	Severity	Client Response	Contributor
LXD-1	`transfer` return value not checked, should use safeTransferFrom()/safeTransfer function instead	Logical	Medium	Fixed	rajatbeladiya, Yaodao
LXD-2	`Ownable` does not implement 2-Step-Process for transferring ownership	Privilege Related	Medium	Acknowledged	0xzoobi, Yaodao
LXD-3	Incorrect logic in `setDistributionRate`	Logical	Medium	Fixed	Yaodao
LXD-4	`Minter.sol` and `VoterV2.sol` Week doesn't represent a WEEK's time	Logical	Low	Fixed	0xWeb3boy
LXD-5	Upgradeable contracts must disable initializers in the implementation contracts	DOS	Low	Acknowledged	0xzoobi

LXD-6	Upgradeable contract is missing a `__gap[50]` storage variable to allow for new storage variables in later versions	Logical	Low	Fixed	rajatbeladiya
LXD-7	Starting weekly emission of 2.4M Loxo instead of 2.6M Loxo	Logical	Informational	Fixed	0xWeb3boy
LXD-8	No need to use SafeMath in solidity version 0.8+	Code Style	Informational	Acknowledged	Yaodao

LXD-1: `transfer` return value not checked, should use safeTransferFrom()/safeTransfer function instead

Category	Severity	Client Response	Contributor
Logical	Medium	Fixed	rajatbeladiya, Yaodao

Code Reference

- code/Loxodrome_Contracts/contracts/MasterChef.sol#L238
- code/Loxodrome_Contracts/contracts/MasterChef.sol#L264

```
238: NFT.transferFrom(msg.sender, address(this), tokenIds[i]);
```

```
264: NFT.transferFrom(address(this), msg.sender, tokenIds[i]);
```

- code/Loxodrome_Contracts/contracts/RewardsDistributor.sol#L293
- code/Loxodrome_Contracts/contracts/RewardsDistributor.sol#L318
- code/Loxodrome_Contracts/contracts/RewardsDistributor.sol#L346

```
293: IERC20(token).transfer(_nftOwner, amount);
```

```
318: IERC20(token).transfer(_nftOwner, amount);
```

```
346: IERC20(_token).transfer(msg.sender, _balance);
```

- target/THENA-Contracts/contracts/RewardsDistributor.sol#L293
- target/THENA-Contracts/contracts/RewardsDistributor.sol#L293
- target/THENA-Contracts/contracts/RewardsDistributor.sol#L318
- target/THENA-Contracts/contracts/RewardsDistributor.sol#L318
- target/THENA-Contracts/contracts/RewardsDistributor.sol#L346
- target/THENA-Contracts/contracts/RewardsDistributor.sol#L346

```
293: IERC20(token).transfer(_nftOwner, amount);
```

```
293: IERC20(token).transfer(_nftOwner, amount);
```

```
318: IERC20(token).transfer(_nftOwner, amount);
```

```
318: IERC20(token).transfer(_nftOwner, amount);
```

```
346: IERC20(_token).transfer(msg.sender, _balance);
```

```
346: IERC20(_token).transfer(msg.sender, _balance);
```

Description

rajatbeladiya: The `` `transferFrom` `` function is used instead of `` `safeTransferFrom` `` and it's discouraged by OpenZeppelin. If the arbitrary address is a contract and is not aware of the incoming ERC721 token, the sent token could be locked.
rajatbeladiya: Some tokens do not implement the ERC20 standard properly but are still accepted by most code that accepts ERC20 tokens. For example Tether (USDT)'s `` `transfer()` `` and `` `transferFrom()` `` functions on L1 do not return booleans as the specification requires, and instead have no return value. When these sorts of tokens are cast to `` `IERC20` ``, their function signatures do not match and therefore the calls made, revert (see this link for a test case).
rajatbeladiya: Not all `` `IERC20` `` implementations `` `revert()` `` when there's a failure in `` `transfer()` ``/`` `transferFrom()` ``. The function signature has a `` `boolean` `` return value and they indicate errors that way instead. By not checking the return value, operations that should have marked as failed, may potentially go through without actually making a payment
Yaodao: The return value of the `` `transfer()` `` call is not checked.

Recommendation

rajatbeladiya: Use `` `safeTransferFrom` `` instead of `` `transferFrom` ``
rajatbeladiya: Use OpenZeppelin's `` `SafeERC20` ``'s `` `safeTransfer()` ``/`` `safeTransferFrom()` `` instead
rajatbeladiya: check return values

```
bool success = IERC20(token).transfer(_nftOwner, amount);
require(success, "transfer failed");
```

Yaodao: Since some ERC-20 tokens return no values and others return a `` `bool` `` value, they should be handled with care. We advise using the OpenZeppelin's `` `SafeERC20.sol` `` implementation to interact with the `` `transfer()` `` and `` `transferFrom()` `` functions of external ERC-20 tokens. The OpenZeppelin implementation checks for the existence of a return value and reverts if `` `false` `` is returned, making it compatible with all ERC-20 token implementations.

Client Response

client response for rajatbeladiya: Fixed on https://github.com/Loxodromexyz/Loxodrome_Contracts/commit/1b38bcdc0696a7e3512944c424fa45957d065304
client response for rajatbeladiya: https://github.com/Loxodromexyz/Loxodrome_Contracts/commit/1b38bcdc0696a7e3512944c424fa45957d065304
client response for rajatbeladiya: Fixed on https://github.com/Loxodromexyz/Loxodrome_Contracts/commit/1b38bcdc0696a7e3512944c424fa45957d065304
client response for Yaodao: Fixed it on https://github.com/Loxodromexyz/Loxodrome_Contracts/commit/4c05e540b3c3bc0befa9eb1f7d02c99d02e6fab6

LXD-2: `Ownable` does not implement 2-Step-Process for transferring ownership

Category	Severity	Client Response	Contributor
Privilege Related	Medium	Acknowledged	0xzoobi, Yaodao

Code Reference

- code/Loxodrome_Contracts/contracts/factories/GaugeFactory.sol#L7

```
7: import "@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol";
```

- code/Loxodrome_Contracts/contracts/MasterChef.sol#L9
- code/Loxodrome_Contracts/contracts/MasterChef.sol#L12

```
9: import "@openzeppelin/contracts/access/Ownable.sol";
```

```
12: contract MasterChef is Ownable {
```

- code/Loxodrome_Contracts/contracts/VoterV2.sol#L21

```
21: contract VoterV2 is IVoter, OwnableUpgradeable, ReentrancyGuardUpgradeable {
```

Description

0xzoobi: To understand the impact of the issue at hand lets reflect on the recent incident involving the `` `$SLERF` `` Solana token, where the developer inadvertently burned the LP tokens. Similarly, if ownership is mistakenly transferred to an incorrect address, it could render all `` `onlyOwner` `` functions inoperative.
Since the privileged roles have critical function roles assigned to them. Assigning the ownership to a wrong user can be disastrous.
So Consider using the `` `Ownable2Step` `` contract from OZ (https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable2Step.sol) instead.
The way it works is there is a `` `transferOwnership` `` to transfer the ownership and `` `acceptOwnership` `` to accept the ownership. Refer the above `` `Ownable2Step.sol` `` for more details.
Yaodao: It is possible that the `` `onlyOnwer` `` role mistakenly transfers ownership to the wrong address, resulting in the loss of the `` `onlyOnwer` `` role.

Recommendation

0xzoobi: Implement 2-Step-Process for transferring ownership via Ownable2Step.
The Fix:

```
- import { Ownable } from "@openzeppelin/contracts/access/Ownable.sol";
+ import { Ownable2Step } from "@openzeppelin/contracts/access/Ownable2Step.sol";
```

Yaodao: Recommend implementing a two-step process where the onwer nominates an account and the nominated account needs to call an **`acceptOnwership()`** function for the transfer of the ownership to fully succeed.

Client Response

client response for 0xzoobi: Acknowledged -
Thank you for bringing this issue to our attention.
We have reviewed the description and recommendation provided for this newly identified issue.
We will update it in the next version.
client response for Yaodao: Acknowledged -
Thank you for bringing this issue to our attention.
We have reviewed the description and recommendation provided for this newly identified issue.
We will update it in the next version.

LXD-3:Incorrect logic in `setDistributionRate`

Category	Severity	Client Response	Contributor
Logical	Medium	Fixed	Yaodao

Code Reference

- code/Loxodrome_Contracts/contracts/MasterChef.sol#L128

```
128: function setDistributionRate(uint256 amount, uint256 amountLOXO) public onlyKeeper {
```

Description

Yaodao: In `MasterChef`, the keeper can `setDistributionRate` to reset the reward rates:

```
function setDistributionRate(uint256 amount, uint256 amountLOXO) public onlyKeeper {
        updatePool();
        uint256 notDistributed;
        if (lastDistributedTime > 0 && block.timestamp < lastDistributedTime) {
            uint256 timeLeft = lastDistributedTime.sub(block.timestamp);
            notDistributed = rewardPerSecond.mul(timeLeft);
        }

        amount = amount.add(notDistributed);
        uint256 _rewardPerSecond = amount.div(distributePeriod);
        rewardPerSecond = _rewardPerSecond;
        lastDistributedTime = block.timestamp.add(distributePeriod);
        amountLOXO = amountLOXO.add(notDistributed);
        uint256 _rewardPerSecondLOXO = amountLOXO.div(distributePeriod);
        rewardPerSecondLOXO = _rewardPerSecondLOXO;
        emit LogRewardPerSecond(_rewardPerSecond, _rewardPerSecondLOXO);
    }
```

There are two reward rates: `rewardPerSecond` and `rewardPerSecondLOXO`. The `rewardPerSecond` is the rate for the token `WIOTX` and the `rewardPerSecondLOXO` is the rate for the token `LOXO`. When calling `setDistributionRate`, it will first calculate the remaining rewards:

```
  if (lastDistributedTime > 0 && block.timestamp < lastDistributedTime) {
            uint256 timeLeft = lastDistributedTime.sub(block.timestamp);
            notDistributed = rewardPerSecond.mul(timeLeft);
        }
```

The `notDistributed` here is the left rewards for token `WIOTX` because it uses the rate `rewardPerSecond`. However, when calculating `amountLOXO`, the `notDistributed` will be added to `amountLOXO`:

```
amountLOXO = amountLOXO.add(notDistributed);
```

It is incorrect to add the number of remaining `WIOTX` tokens to the number of `LOXO` tokens

Recommendation

Yaodao: Recommend calculating the remaining amount of the `LOXO` token and adding it to the `amountLOXO`:

```
  if (lastDistributedTime > 0 && block.timestamp < lastDistributedTime) {
            uint256 timeLeft = lastDistributedTime.sub(block.timestamp);
            notDistributed = rewardPerSecond.mul(timeLeft);
            notDistributedLoxo = rewardPerSecondLOXO.mul(timeLeft);

        }


…
…
amountLOXO = amountLOXO.add(notDistributedLoxo);
```

Client Response

client response for Yaodao: Fixed - the address is https://github.com/Loxodromexyz/Loxodrome_Contracts/commit/5c5d8600fd0d6b3ad4b47d13270b66de6368404f

LXD-4: `Minter.sol` and `VoterV2.sol` Week doesn't represent a WEEK's time

Category	Severity	Client Response	Contributor
Logical	Low	Fixed	0xWeb3boy

Code Reference

- code/Loxodrome_Contracts/contracts/Minter.sol#L29

```
29: uint internal constant WEEK = 86400 * 2; // allows minting once per week (reset every Thursday 00:00 UTC)
```

- code/Loxodrome_Contracts/contracts/VoterV2.sol#L28

```
28: uint internal constant DURATION = 2 days; // rewards are released over 7 days
```

Description

0xWeb3boy: There are various instances where **`` `uint internal constant WEEK = 86400 * 2;` ``** where it should have been **`` `uint internal constant WEEK = 86400 * 7;` ``** since a the natspec says **`` `allows minting once per week (reset every Thursday 00:00 UTC)` ``** that means a week should have 7 days and this constant must represent a unit and hence prevents minting more than once a week.
Hoewver the forked code changes it to 2 which means a week have now only two days, which further means that it will only allow to mint once every 2 days, Now there are various instances where.
https://github.com/Secure3Audit/code_Loxodrome/blob/main/code/Loxodrome_Contracts/contracts/Minter.sol#L29
I suppose that the Loxo's design allows minting once in every two week, that is why everywhere the code has a cycle of 2 days instead of 7.
Now this could be a medium or also can be an low/informational based on how the protocol has designed its each cycle to be.
If the cycle is supposed to be of 7 days it is of Medium severity and if the cycle is suuposed to be of 2 days it will be of Low severity.

Recommendation

0xWeb3boy: change the code to the following code :

```
- uint internal constant WEEK = 86400 * 2;
+ uint internal constant WEEK = 86400 * 7;
```

Either change the comment or change the duration to 7 days

```
-    uint internal constant DURATION = 7 days; // rewards are released over 7 days
+    uint internal constant DURATION = 2 days; // rewards are released over 2 days
```

Client Response

client response for 0xWeb3boy: Fixed - This is fixed on branch https://github.com/Loxodromexyz/Loxodrome_Contracts/tree/develop

LXD-5:Upgradeable contracts must disable initializers in the implementation contracts

Category	Severity	Client Response	Contributor
DOS	Low	Acknowledged	0xzoobi

Code Reference

- code/Loxodrome_Contracts/contracts/factories/GaugeFactory.sol#L16

```
16: constructor() {}
```

Description

0xzoobi: Quoting OpenZeppelin

```
Avoid leaving a contract uninitialized.

An uninitialized contract can be taken over by an attacker. This applies to both a proxy and its i
mplemen-
tation contract, which may impact the proxy. To prevent the implementation contract from being use
d,
you should invoke the _disableInitializers function in the constructor to automatically lock it wh
en it
is deployed.
```

Similar reports from past audits

1. https://github.com/solodit/solodit_content/blob/main/reports/Pashov/2023-08-01-gTrade.md#l-01-implementation-contract-can-be-initialized
2. https://github.com/mixbytes/audits_public/blob/master/Aspida%20Network/README.md#3-constructors-may-use-_disableinitializers-instead-of-initialization-routine

Recommendation

0xzoobi: Add the following code to the GaugeFactory contract.

```
constructor() {
  _disableInitializers();
}
```

Client Response

client response for 0xzoobi: Acknowledged -

LXD-6:Upgradeable contract is missing a `__gap[50]` storage variable to allow for new storage variables in later versions

Category	Severity	Client Response	Contributor
Logical	Low	Fixed	rajatbeladiya

Code Reference

- code/Loxodrome_Contracts/contracts/VoterV2.sol#L17-L21

```
17: import "@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol";
18: import "@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol";
19:
20:
21: contract VoterV2 is IVoter, OwnableUpgradeable, ReentrancyGuardUpgradeable {
```

Description

rajatbeladiya: See https://docs.openzeppelin.com/contracts/4.x/upgradeable#storage_gaps link for a description of this storage variable.
The contract VoterV2 is designed to be upgradeable using OpenZeppelin's upgradeable contracts. However, it is missing a __gap[50] storage variable. This variable is crucial for the upgradeability feature as it reserves space in storage for future variables that might be added in later versions of the contract.

Recommendation

rajatbeladiya: Add a __gap[50] storage variable at the end of the contract's storage declarations. This will reserve the necessary space in storage for future upgrades.

```
uint256[50] private __gap;
```

Client Response

client response for rajatbeladiya: Fixed - the address is https://github.com/Loxodromexyz/Loxodrome_Contracts/commit/187a709d19a8f26f283abe8ece562b9fe1bf2c8e

LXD-7:Starting weekly emission of 2.4M Loxo instead of 2.6M Loxo

Category	Severity	Client Response	Contributor
Logical	Informational	Fixed	0xWeb3boy

Code Reference

- code/Loxodrome_Contracts/contracts/Minter.sol#L31

```
31: uint public weekly = 2_400_000 * 1e18; // represents a starting weekly emission of 2.6M Loxo (Lo
xo has 18 decimals)
```

Description

0xWeb3boy: The Natspec in the code says the initial weekly emission is set to 2.6M Loxo where in the code it is set to `` `2.4M` `` Loxo.

```
uint public weekly = 2_400_000 * 1e18;
```

Recommendation

0xWeb3boy: change the code to following

```
- uint public weekly = 2_400_000 * 1e18;
+ uint public weekly = 2_600_000 * 1e18;
```

Client Response

client response for 0xWeb3boy: Fixed - Fixed to 0.375m on https://github.com/Loxodromexyz/Loxodrome_Contracts/commit/14e8a1edd1a4a9830b5ba6bdecf0401b2a96fd20

LXD-8:No need to use SafeMath in solidity version 0.8+

Category	Severity	Client Response	Contributor
Code Style	Informational	Acknowledged	Yaodao

Code Reference

- code/Loxodrome_Contracts/contracts/MasterChef.sol#L13

```
13: using SafeMath for uint256;
```

Description

Yaodao: Solidity provides overflow checking for version above 0.8. The contract does not need to import SafeMath library for overflow checking, which can save gas.

Recommendation

Yaodao: Recommend removing SafeMath library.

Client Response

client response for Yaodao: Acknowledged -

Disclaimer