Competitive Security Assessment

Lymex Token

Nov 11th, 2022

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 Detail

Project Name	Lymex Token
Platform & Language	Solidity
Codebase	• audit code address - https://bscscan.com/address/0x448F861728F4Ed28914f42B4C8C0ff70b8bD637B#code • final code address - Not Provided
Audit Methodology	• Audit Contest • Business Logic and Code Review • Privileged Roles Review • Static Analysis

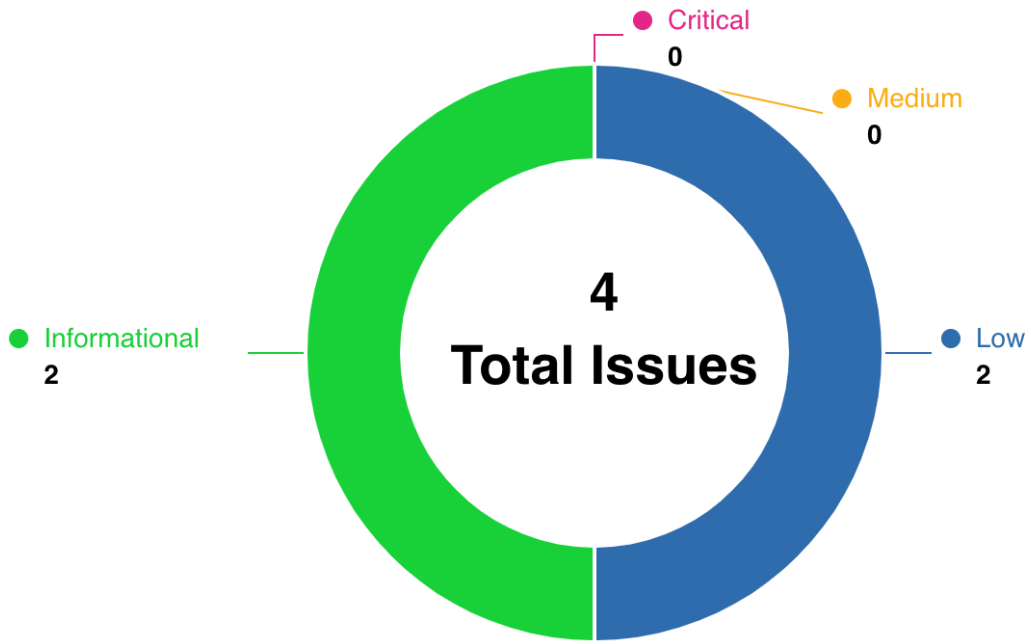
Code Vulnerability Review Summary

Vulnerability Level	Total	Reported	Acknowledged	Fixed	Mitigated	Declined
Critical	0	0	0	0	0	0
Medium	0	0	0	0	0	0
Low	2	0	2	0	0	0
Informational	2	0	2	0	0	0

Audit Scope

File	Commit Hash
lym_token.sol	b31bf0249f901e1384bcd9f9b5f1049eb5c7fec4e0cd496a28ec849b4c5b9336

Code Assessment Findings



ID	Name	Category	Severity	Status	Contributor
LYM-1	`constructor` does not validate the input address parameters	Logical	Low	Acknowledged	Secure3
LYM-2	Miss emitting event in important state-changing setter functions	Code Style	Informational	Acknowledged	Secure3
LYM-3	`transactionFee` should be capped	Privilege Related	Low	Acknowledged	Secure3
LYM-4	Lack of precision in the calculation of fee	Logical	Informational	Acknowledged	Secure3

LYM-1: `constructor` does not validate the input address parameters

Category	Severity	Code Reference	Status	Contributor
Logical	Low	• code/contracts/project/lym/lym_token.sol#L38-L39	Acknowledged	Secure3

Code

```
38:         address router_,
39:         address usdt_,
```

Description

Secure3 : The input parameter `address router_` and `address usdt_` is not validated in the contructor and there is no setter functions to update them in case the value is not passed correctly. If the value is not passed correctly, owner has to redeploy the contract.

Recommendation

Secure3 : Validate the input address parameter.

```
require(address(router_) != address(0));
require(address(usdt_) != address(0));
```

Client Response

Acknowledged

LYM-2:Miss emitting event in important state-changing setter functions

Category	Severity	Code Reference	Status	Contributor
Code Style	Informational	• code/contracts/project/lym/lym_token.sol#L72-L95	Acknowledged	Secure3

Code

```
72:     function setContractStatus(bool status_) public onlyOwner {
73:         meta.whiteContract = status_;
74:     }
75:
76:     function setAutoAddLiquidity(bool status_) public onlyOwner {
77:         meta.autoAddLiquidity = status_;
78:     }
79:
80:     function setLiquidityHelper(address helper_) public onlyOwner {
81:         meta.helper = IPancakeHelper(helper_);
82:         whiteList[helper_] = true;
83:         whiteContracts[helper_] = true;
84:     }
85:
86:     function setMiningPool(address pool_) public onlyOwner {
87:         meta.miningPool = ILiquidityPool(pool_);
88:         whiteList[pool_] = true;
89:         whiteContracts[pool_] = true;
90:     }
91:
92:     function setFee(uint256 transactionFee_, uint256 addLiquidityAmount_) public onlyOwner {
93:         meta.transactionFee = transactionFee_;
94:         meta.addLiquidityAmount = addLiquidityAmount_;
95:     }
```

Description

Secure3 : Miss emitting event in important state-changing setter functions: `setContractStatus()`, `setAutoAddLiquidity()`, `setLiquidityHelper()`, `setMiningPool()`, `setFee()`.

Recommendation

Secure3 : Emit event at the end of the important setter functions.

```
emit ContractStatusSet(...);
emit AutoAddLiquiditySet(...);
emit LiquidityHelperSet(...);
emit MiningPoolSet(...);
emit FeeSet(...);
```

Client Response

Acknowledged

LYM-3: `transactionFee` should be capped

Category	Severity	Code Reference	Status	Contributor
Privilege Related	Low	• code/contracts/project/lym/lym_token.sol#L165-L172	Acknowledged	Secure3

Code

```
165:                uint256 fee = (amount * meta.transactionFee) / 100;
166:                _transfer(from, address(meta.miningPool), fee / 2);
167:                _transfer(from, address(meta.helper), fee / 2);
168:
169:                unchecked {
170:                    meta.miningPool.addReward(fee / 2);
171:                }
172:                amount -= fee;
```

Description

Secure3 : The transaction fee is calculated as `uint256 fee = (amount * meta.transactionFee) / 100;` and then `amount -= fee;` is deducted from the transfer `amount` in the `_processTransfer()` function. The fee itself is distributed half and half to the `meta.miningPool` and `meta.helper`.
While technically the fee can be set between [0,100], to prevent the fee is charged too high, the project owner should cap the fee at a reasonable percentage and make it transparent to the community.

Recommendation

Secure3 : Consider below fix

```
    function setFee(uint256 transactionFee_, uint256 addLiquidityAmount_) public onlyOwner {
        require(transactionFee_ < xx, "fee too high");
        meta.transactionFee = transactionFee_;
        meta.addLiquidityAmount = addLiquidityAmount_;
    }
```

Client Response

Acknowledged

LYM-4:Lack of precision in the calculation of fee

Category	Severity	Code Reference	Status	Contributor
Logical	Informational	• code/contracts/project/lym/lym_token.sol#L165-L167	Acknowledged	Secure3

Code

```
165:                uint256 fee = (amount * meta.transactionFee) / 100;
166:                _transfer(from, address(meta.miningPool), fee / 2);
167:                _transfer(from, address(meta.helper), fee / 2);
```

Description

Secure3 : The transaction fee is divided into two parts. Use subtraction to avoid the precision issue.

Recommendation

Secure3 : Consider below fix in the `_processTransfer` function

```
uint256 fee = (amount * meta.transactionFee) / 100;
_transfer(from, address(meta.miningPool), fee / 2);
_transfer(from, address(meta.helper), fee - fee / 2);
```

Client Response

Acknowledged

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Invoices, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Invoice. This report provided in connection with the services set forth in the Invoices shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Invoice. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without Secure3’s prior written consent in each instance.

This report is not an “endorsement” or “disapproval” of any particular project or team. This report is not an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Secure3 to perform a security assessment. This report does not provide any warranty or guarantee of free of bug of codes analyzed, nor do they provide any indication of the technologies, business model or legal compliancy.

This report should not be used in any way to make decisions around investment or involvement with any particular project. Instead, it represents an extensive assessing process intending to help our customers increase the quality of their code and high-level consistency of implementation and business model, while reducing the risk presented by cryptographic tokens and blockchain technology.

Secure3’s position on the final decisions over blockchain technologies and corresponding associated transactions is that each company and individual are responsible for their own due diligence and continuous security.

The assessment services provided by Secure3 is subject to dependencies and under continuing development. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.