soken

SMART CONTRACT
SECURITY AUDIT

MarsRise

August, 2021

Table of Contents

Disclaimer

This is a comprehensive report based on our automated and manual examination of cybersecurity vulnerabilities and framework flaws. We took into consideration smart contract based algorithms, as well. Reading the full analysis report is essential to build your understanding of project's security level. It is crucial to take note, though we have done our best to perform this analysis and report, that you should not rely on the our research and cannot claim what it states or how we created it. Before making any judgments, you have to conduct your own independent research. We will discuss this in more depth in the following disclaimer - please read it fully.

DISCLAIMER: You agree to the terms of this disclaimer by reading this report or any portion thereof. Please stop reading this report and remove and delete any copies of this report that you download and/or print if you do not agree to these conditions. This report is for non-reliability information only and does not represent investment advice. No one shall be entitled to depend on the report or its contents, and Soken and its affiliates shall not be held responsible to you or anyone else, nor shall Soken provide any guarantee or representation to any person with regard to the accuracy or integrity of the report. Without any terms, warranties or other conditions other than as set forth in that exclusion and Soken excludes hereby all representations, warrants, conditions and other terms (including, without limitation, guarantees implied by the law of satisfactory quality, fitness for purposes and the use of reasonable care and skills). The report is provided as "as is" and does not contain any terms and conditions. Except as legally banned, Soken disclaims all responsibility and responsibilities and no claim against Soken is made to any amount or type of loss or damages (without limitation, direct, indirect, special, punitive, consequential or pure economic loses or losses) that may be caused by you or any other person, or any damages or damages, including without limitations (whether innocent or negligent).

Security analysis is based only on the smart contracts. No applications or operations were reviewed for security. No product code has been reviewed.

Procedure

Our analysis contains following steps:

1. Project Analysis;

2. Manual analysis of smart contracts:
- Deploying smart contracts on any of the network(Ropsten/Rinkeby) using Remix IDE
- Hashes of all transaction will be recorded
- Behaviour of functions and gas consumption is noted, as well.

3. Unit Testing:
- Smart contract functions will be unit tested on multiple parameters and under multiple conditions to ensure that all paths of functions are functioning as intended.
- In this phase intended behaviour of smart contract is verified.
- In this phase, we would also ensure that smart contract functions are not consuming unnecessary gas.
- Gas limits of functions will be verified in this stage.

4. Automated Testing:
- Mythril
- Oyente
- Manticore
- Solgraph

Terminology

We categorize the finding into 4 categories based on their vulnerability:

- Low-severity issue — less important, must be analyzed
- Medium-severity issue — important, needs to be analyzed and fixed
- High-severity issue —important, might cause vulnerabilities, must be analyzed and fixed
- Critical-severity issue —serious bug causes, must be analyzed and fixed.

Limitations

The security audit of Smart Contract cannot cover all vulnerabilities. Even if no vulnerabilities are detected in the audit, there is no guarantee that future smart contracts are safe. Smart contracts are in most cases safeguarded against specific sorts of attacks. In order to find as many flaws as possible, we carried out a comprehensive smart contract audit. Audit is a document that is not legally binding and guarantees nothing.

Token Contract Details for 28.08.2021

Contract Name: **MARSRISE**

Deployer address: **0x184079Ca987F562ae6a0c59f4BE5cADB20323863**

Total Supply: **1,000,000,000,000,000**

Token Tracker: **MARSRISE**

Decimals: **9**

Token holders: **147**

Transactions count: **558**

Top 100 holders dominance: **99.61%**

Contract deployer address:
0x184079Ca987F562ae6a0c59f4BE5cADB20323863

Audit Details

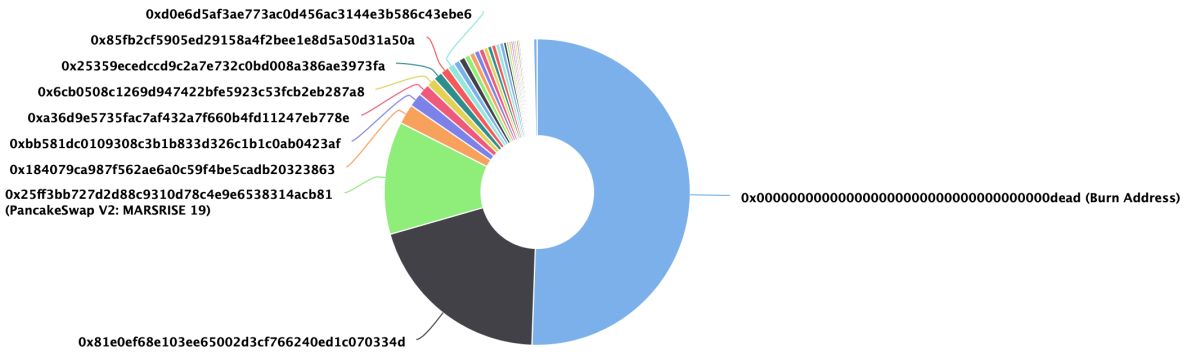
Project Name: **MarsRise**

Language: **Solidity**

Blockchain: **BSC**

Project Website: **marsrise.net**

MarsRise Token Distribution



MarsRise Top 10 Holders

Rank	Address	Quantity (Token)	Percentage
1	Burn Address	505,613,414,475,512.335902135	50.5613%
2	0x81e0ef68e103ee65002d3cf766240ed1c070334d	200,000,000,000,000	20.0000%
3	PancakeSwap V2: MARSRISE 19	119,157,210,394,497.008789656	11.9157%
4	0x184079ca987f562ae6a0c59f4be5cadb20323863	20,918,116,522,026.165010938	2.0918%
5	0xbb581dc0109308c3b1b833d326c1b1c0ab0423af	14,810,808,284,490.733359408	1.4811%
6	0xa36d9e5735fac7af432a7f660b4fd11247eb778e	12,390,677,076,996.776841032	1.2391%
7	0x6cb0508c1269d947422bfe5923c53fcb2eb287a8	9,642,946,384,296.710479323	0.9643%
8	0x25359ecedccd9c2a7e732c0bd008a386ae3973fa	9,525,365,771,146.516483213	0.9525%
9	0x85fb2cf5905ed29158a4f2bee1e8d5a50d31a50a	8,701,059,389,099.395881167	0.8701%
10	0xd0e6d5af3ae773ac0d456ac3144e3b586c43ebe6	7,395,787,417,212.531120103	0.7396%

Vulnerabilities checking

Issue Description	Checking Status
Compiler Errors	Completed
Delays in Data Delivery	Completed
Re-entrancy	Completed
Transaction-Ordering Dependence	Completed
Timestamp Dependence	Completed
Shadowing State Variables	Completed
DoS with Failed Call	Completed
DoS with Block Gas Limit	Low-issues
Outdated Complier Version	Completed
Assert Violation	Completed
Use of Deprecated Solidity Functions	Completed
Integer Overflow and Underflow	Completed
Function Default Visibility	Completed
Malicious Event Log	Completed
Math Accuracy	Completed
Design Logic	Completed
Fallback Function Security	Completed
Cross-function Race Conditions	Completed
Safe Zeppelin Module	Completed

Security Issues

1) Out of Gas issue:

The function includeInRewards() uses the loop to find and remove addresses from the _excluded list. Function will be aborted with OUT_OF_GAS exception if there will be a long excluded addresses list.

```
628     function includeInReward(address account) external onlyOwner() {
629         require(_isExcluded[account], "Account is already excluded");
630         for (uint256 i = 0; i < _excluded.length; i++) {
631             if (_excluded[i] == account) {
632                 _excluded[i] = _excluded[_excluded.length - 1];
633                 _tOwned[account] = 0;
634                 _isExcluded[account] = false;
635                 _excluded.pop();
636                 break;
637             }
638         }
639     }
```

2) Out of Gas issue:

The function _getCurrentSupply also uses the loop for evaluating total supply. It also could be aborted with OUT_OF_GAS exception if there will be a long excluded addresses list.

```
848     function _getCurrentSupply() private view returns(uint256, uint256) {
849         uint256 rSupply = _rTotal;
850         uint256 tSupply = _tTotal;
851         for (uint256 i = 0; i < _excluded.length; i++) {
852             if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return (_rTotal, _tTotal);
853             rSupply = rSupply.sub(_rOwned[_excluded[i]]);
854             tSupply = tSupply.sub(_tOwned[_excluded[i]]);
855         }
856         if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
857         return (rSupply, tSupply);
858     }
```

Recommendation:

Use EnumerableSet instead of array or do not use long arrays.

Conclusion

Low-severity issues exist within smart contracts. Smart contracts are free from any critical or high-severity issues.

NOTE: Please check the disclaimer above and note, that audit makes no statements or warranties on business model, investment attractiveness or code sustainability.