soken

SMART CONTRACT SECURITY AUDIT

Masrelic

Scan and check this report was posted at Soken Github

August, 2022

Website: soken.io

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.

Basic Security Recommendation

Unlike hardware and paper wallets, hot wallets are connected to the internet and store private keys online, which exposes them to greater risk. If a company or an individual holds significant amounts of cryptocurrency in a hot wallet, they should consider using MultiSig addresses. Wallet security is enhanced when private keys are stored in different locations and are not controlled by a single entity.

More info: **https://medium.com/coinmonks/guide-to-using-the-gnosis-multisig-wallet-eth-e76979741162**

Token Contract Details for 24.08.2022

Contract Name: **MasRelic**

Deployed address: **0x77132Bb08B02a7a87732ec289EFaB45edF49884F**

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

Token Tracker: **RELIC**

Decimals: **18**

Token holders: **61**

Transactions count: **1410**

Top 100 holders dominance: **100.00%**

Audit Details

Project Name: **Masrelic**

Language: **Solidity**

Compiler Version: **v0.8.10**

Blockchain: **Ethereum**

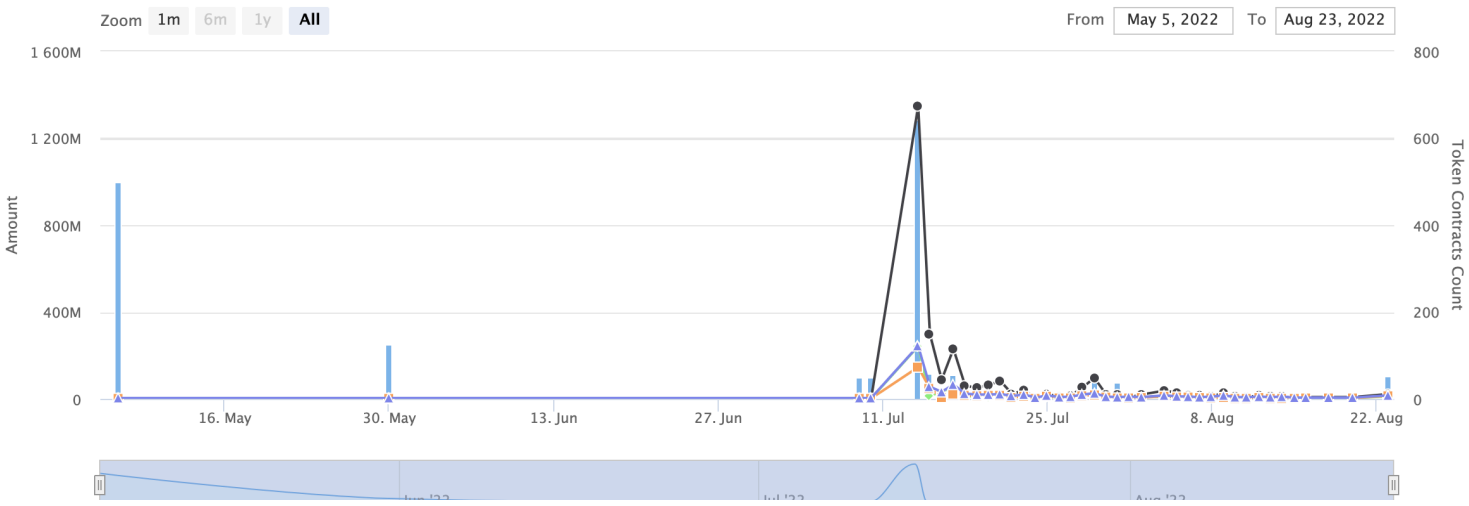
Social Profiles

Project Website: **https://www.masrelic.com/**

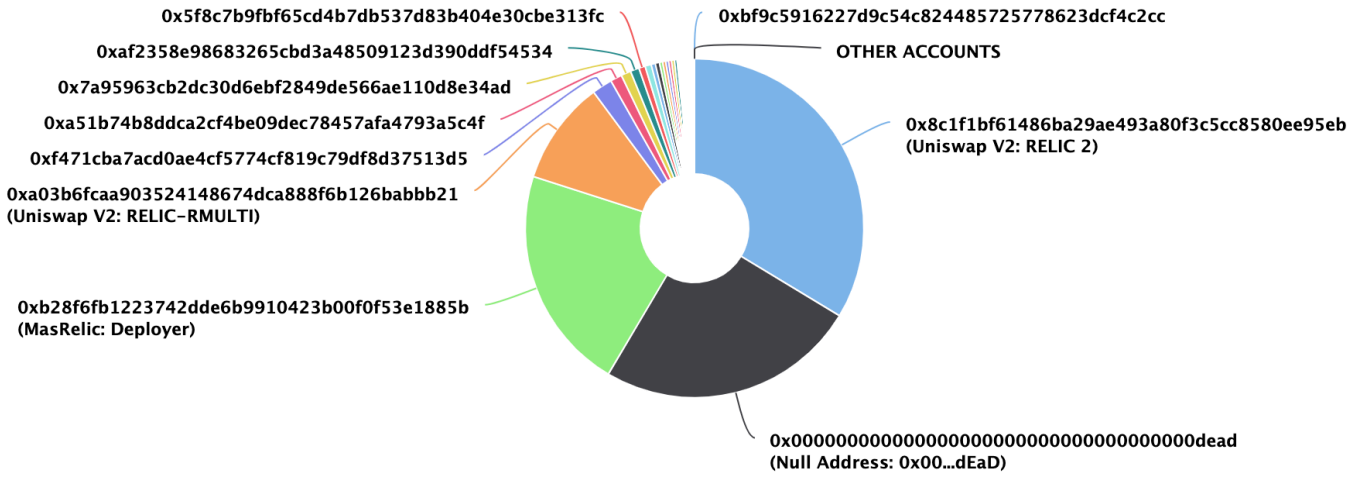
Project Twitter: **https://twitter.com/MasRelic**

Project Telegram: **https://t.me/MasRelic_EN**

Token Analytics



RELIC Token Distribution



RELIC Top 10 Holders

Rank	Address	Quantity (Token)	Percentage
1	Uniswap V2: RELIC 2	336,360,156.79178578127417926	33.6360%
2	Null Address: 0x00...dEaD	248,885,502	24.8886%
3	MasRelic: Deployer	214,184,745.183324737819626535	21.4185%
4	Uniswap V2: RELIC-RMULTI	99,000,353.553390593273762205	9.9000%
5	0xf471cba7acd0ae4cf5774cf819c79df8d37513d5	19,370,064.269989289231684316	1.9370%
6	0xa51b74b8ddca2cf4be09dec78457afa4793a5c4f	11,121,727.247874617555765969	1.1122%
7	0x7a95963cb2dc30d6ebf2849de566ae110d8e34ad	9,405,556.931725723834765437	0.9406%
8	0xaf2358e98683265cbd3a48509123d390ddf54534	8,480,564.704857208345114782	0.8481%
9	0x5f8c7b9fbf65cd4b7db537d83b404e30cbe313fc	6,052,093.785520312843992968	0.6052%
10	0x290420874bf65691b98b67d042c06bbc31f85f11	5,786,393.20986099010100503	0.5786%

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	Completed
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) Volatile Code:

The return values of functions *swapExactTokensForETHSupportingFeeOnTransferTokens* and *addLiquidityETH* are not properly handled.

Recommendation:

We recommend using variables to receive the return value of the functions mentioned above and handle both success and failure cases if needed by the business logic.

2) Owner Privileges

The contract contains ownership functionality and ownership is not renounced which allows the creator or current owner to modify contract behaviour (for example, mint new tokens, modify tx fee).

3) Custom errors to save gas: L2281, L2290, L2294, L2302

```
2280        function transfer(address, uint256) public pure override returns (bool) {
2281            revert("Mas_DividendTracker: method not implemented");
2282        }
2283
2284        function allowance(address, address)
2285            public
2286            pure
2287            override
2288            returns (uint256)
2289        {
2290            revert("Mas_DividendTracker: method not implemented");
2291        }
2292
2293        function approve(address, uint256) public pure override returns (bool) {
2294            revert("Mas_DividendTracker: method not implemented");
2295        }
2296
```

The contract was found to be using revert() statements. Since Solidity v0.8.4, custom errors have been introduced which are a better alternative to the revert. This allows the developers to pass custom errors with dynamic data while reverting the transaction and also making the whole implementation a bit cheaper than using reverts.

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.

Soken Contact Info

Website: www.soken.io

Mob: (+1)416-875-4174

32 Britain Street, Toronto, Ontario, Canada

Telegram: @team_soken

GitHub: sokenteam

Twitter: @soken_team