HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Membrana
Date: December 9, 2018
Platform: Ethereum
Language: Solidity

Document

Name	Smart Contract Code Secondary Review and Security Analysis Report for Membrana
Platform	Ethereum / Solidity
Link	https://github.com/membrana-io/token
Commit	414d8cb676f286a31d1f76311a872c1ecbb5b279
Date	09.12.2018

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Membrana (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer`s smart contract and its code review on December 9th, 2018.

Scope

The scope of the project is **Membrana** smart contracts, which can be found on Github by the link below:

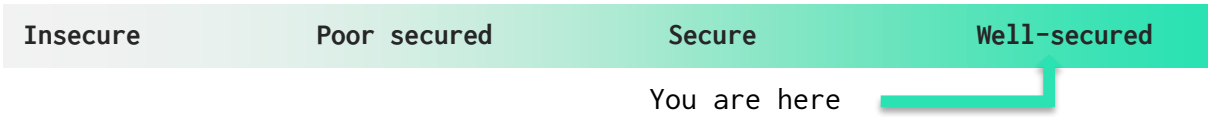
https://github.com/membrana-io/token

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered (the full list includes them but is not limited to them):

- Reentrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Byte array vulnerabilities
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Unchecked external call - Unchecked math
- Unsafe type inference
- Implicit visibility level

Executive Summary

According to the assessment, Customer`s smart contracts are well-secured.



Our team performed analysis of code functionality, manual audit and automated checks with Mythril, Slither and remix IDE (see Appendix A pic 1-5). All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in Audit overview section. General overview is presented in AS-IS section and all found issues can be found in Audit overview section.

We found 0 vulnerabilities in smart contracts. We also outline 2 code style issues.

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens lose etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

AS-IS overview

Token contract overview

Token contract implements **IToken** interface and describes custom ERC20 token with next parameters:

- **name** - **Membrana**
- **symbol** - **MBN**
- **decimals** - **18**

Token has 3 modifiers:

- **controllerOnly** - checks whether **msg.sender** is **controller**.
- **releasedOnly** - checks whether **isReleased** is **true**.
- **notReleasedOnly** - checks whether **isReleased** is **false**.

Token has 8 functions:

- **mint** is a public function - mints specified amount of tokens to specified address. Has **controllerOnly** and **notReleasedOnly** modifiers.
- **transfer** is a public function - transfers specified amount of tokens to specified address. Has **releasedOnly** modifier.
- **transferFrom** is a public function - transfers specified amount of tokens from one specified address to another. Has **releasedOnly** modifier.
- **approve** is a public function - approves specified amount of tokens to specified address. Has **releasedOnly** modifier.

- **increaseAllowance** is a public function - increases amount of allowed tokens for specified address. Has **releasedOnly** modifier.
- **decreaseAllowance** is a public function - decreases amount of allowed token for specified address. Has **releasedOnly** modifier.
- **release** is a public function - changes **isReleased** to **true**. Has **controllerOnly** and **notReleasedOnly** modifiers.
- **setController** is a public function - changes **controller** to specified address. Has **controllerOnly** modifier.

Audit overview

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

No low severity vulnerabilities were found.

Lowest / Code style / Best Practice

Code style

1. Comma must be separated from next element by space on line 59.
2. Use double quotes for string literals on lines: 3, 6, 7, 24, 25, 30, 35, 59, 104.

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract high level description of functionality was presented in As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Overall quality of reviewed contracts is good.

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

The audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on blockchain platform. The platform, its programming language, and other software related to the smart contract can have own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

Appendix A. Automated tools reports

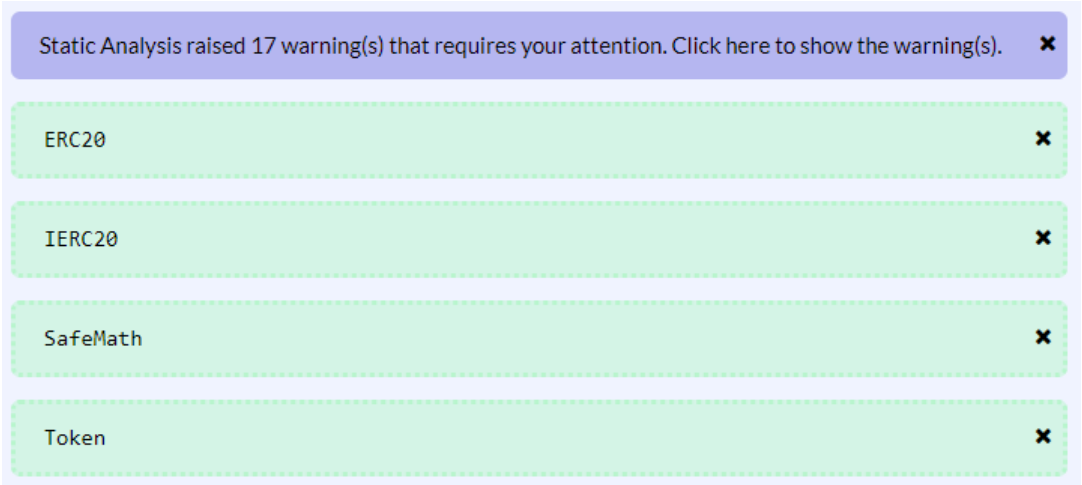
Pic 1. Slither automated report :

```
max@Hacken:~/solidity/projects/Membrana/contracts$ slither Token.sol
INFO:Detectors:Different version of Solidity used in Token.sol: ['^0.4.24', '0.4.25']
INFO:Detectors:Old version of Solidity used in Token.sol: ['0.4.25']
INFO:Slither:Token.sol analyzed (8 contracts), 2 result(s) found
max@Hacken:~/solidity/projects/Membrana/contracts$
```

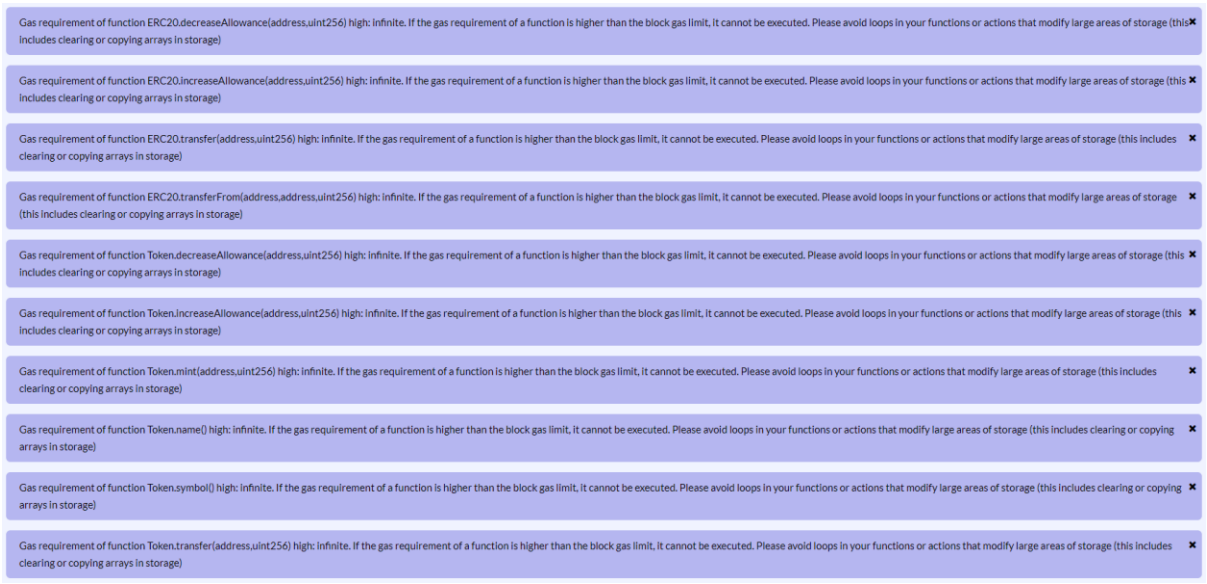
Pic 2. Mythril automated report:

```
max@Hacken:~/solidity/projects/Membrana/contracts$ myth -x Token.sol
==== Integer Overflow ====
Type: Warning
Contract: Token
Function name: transferFrom(address,address,uint256)
PC address: 7056
A possible integer overflow exists in the function `transferFrom(address,address,uint256)`.
The addition or multiplication may result in a value higher than the maximum representable integer.
--------------------
In file: SafeMath.sol:51

a + b

--------------------


max@Hacken:~/solidity/projects/Membrana/contracts$
```

Pic 3. Remix IDE automated report part 1:



Pic 4. Remix IDE automated report part 2:



Pic 5. Remix IDE automated report part 3: