SMART NATION CERTIFICATIONS

UEN: 201536369M

43 Hindhede Walk, #08-04, South Haven I, Singapore 587973

MORPHEUS LABS ERC20 TOKEN SMART CONTRACT

AUDIT REPORT

Reference: SNC\Singapore\00030182\2018

Smart contract testing Methodology

a) **Inspection element-** Examine Smart Contract Token code as uploaded on GitHub on 25 March 2018 in accordance with security baselines of Open Zeppelin smart contract framework guidance and ISO 27001:2013 standard for specific to Application security.

b) **Analysis and evaluation of the Environment**: Review existing Code for adequacy against TVA and existing Controls as required by Morpheus Labs. Review of existing syntax and workflow by compiling code and running our Penetration test tools.

c) **Inspection Report –**The results of this analysis documented within this report.

Overview of Smart Contract code of Morpheus Labs

1. Framework used in the Smart Contract

   Open Zeppelin Smart Contract Framework

2. Tokens used within the Smart Contract

   ERC20 Token

3. Other recommendations for action

   As in audit report below

   Audit Summary: Smart Contracts for ERC20 token of Morpheus Labs do not have any issues of high or critical severity.

Auditors Involved

The following Smart Nation Certifications Pte Ltd personnel will be involved in this security assessment.

1. Indranil Mukherjee, Lead Auditor, Chief Information Security Officer, Smart Nation Certifications Pte Ltd Phone: +65 94577343

Sample profile on https://www.rsaconference.com/speakers/indranil-mukherjee

2. Isha Agarwal
3. Pinaz Anant Patankar
4. Mickey Johnysson

Audit Findings

The audited code is located in the GitHub-token-distribution repository.

The source file version used for this report is Pragma solidity 0.4.17 .

Here is our assessment and recommendations, in order of importance.

Critical Severity

No issues of critical severity.

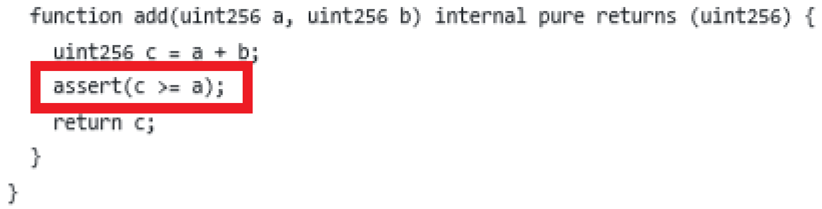
High Severity

No issues of high severity.

Medium Severity

Possible overflow in assert statement that could introduce vulnerabilities

If a is zero (0), then the assert statement would throw an error. It might be quicker to have an IF-ELSE statement to first check for a=0 else return a*b
same with div, check if b=0 else return a/b



```
  function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    assert(c >= a);
    return c;
  }
}
```

Low Severity

Additional ERC20 Interfaces

The IERC20 contract defines the basic interface of a standard token to be used by the Morpheus contract. Whilst this contract does follow the ERC20 standard , the number of require functions need to align with the ERC20 contract from the OpenZeppelin library.

Test Scenario: Consider if the requirement is not met and it returns an error. There may be no way of dealing with such an error.

Other Recommendations

1. There are 2 functions to transfer token from one address to another. They could be combined to reduce the number of steps the compiler needs to take, whilst ensuring that there is NO transfer of tokens in the function transferTokens whose return value is false.

2. Consider the installation of Morpheus ERC20 Token/ OpenZeppelin via NPM, which is the recommended way to use OpenZeppelin smart contracts, which is via the zeppelin-solidity NPM package, allowing for any bugfixes to be easily integrated into the codebase.

3. Consider prohibiting the null address as a parameter of the Morpheus ERC20 Token constructor.

Audit Conclusion

There are no critical or high severity issues in the Morpheus ERC20 token which is overall in compliance with Open Zeppelin Smart contract framework guidelines.

NOTE: Some changes were proposed to follow best practices and reduce potential attack surface.

Yours Sincerely,

Signed: Mukherjee
For Smart Nation Certifications Pte Ltd **Indranil Mukherjee** **Email: indy@sncerts.com**
Date of Signature: 27th March 2018
Position: Chief Information Security Officer