Defence Group_
CYBERSECURITY

REPORT
OTN TOKEN SMART CONTRACT AUDIT RESULTS
FOR THE OTN FOUNDATION

Change history

Version	Date	Author	Changes
1.0	27.09.2017	Defence Group	Report created

1 Contents

2 Acronyms and Abbreviations

ETHEREUM – An open source platform for creating decentralized online services based on the blockchain (Dapps or Decentralized applications) that operate using smart contracts.
ETH (ether) – The cryptocurrency and token in the Ethereum blockchain that is used for payment of transactions and computing services in the Ethereum network.
SOLIDITY – An object-oriented programming language for creating self-fulfilling contracts for the Ethereum platform.
SMART CONTRACT – A computer algorithm designed to create and support contracts in the blockchain technology.
ERC20 – The Ethereum token standard used for Ethereum smart contracts. It is a set of rules for the implementation of Ethereum tokens.
SAFEMATH – A Solidity library created for secure mathematical operations.

3 Introduction

3.1 Vulnerability Level

• **Low severity** – A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
• **Medium severity** – A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
• **High severity** – A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
• **Critical severity** – A vulnerability that can disrupt the contract's functioning in a number of scenarios, or creates a risk that the contract may be broken.

4 Main Results

4.1 SafeMath.sol Evaluation

This library meets modern security requirements and does not require changes.

4.2 Shareable.sol Evaluation

4.2.1 **Low Severity**

o The "*sha3*" function has been deprecated, and now the "keccak256" function is used instead. However, the "*sha3*" function is currently an alias of the "*keccak256*" function.

• Recommendation: Replace the "*sha3*" function with the "*keccak256*" function.

o The "*addressNotNull*" modifier checks the address for a null value, but it compares the address data type with a null value. At this time, implicit type conversion is being performed by the compiler, but we recommend to do this explicitly.

• Recommendation: Before 0, add an explicit conversion to the address type (**address**(0)).

o The "*validRequirement*" modifier checks two values and has three conditions. However, the "*_ownersCount > 0*" condition is not necessary, since the "*_ownersCount > = _required*" check is performed later, but the condition "*_required > 1*" is useful.

• Recommendation: Remove the unnecessary condition "*_ownersCount > 0*".

o Due to the fact that the function of the "*Shareable*" constructor is only invoked once when the contract is deployed, addresses could be checked before the contract is initialized. In this case, the checks inside the loop are not needed. If the checks are necessary, we recommend explicitly defining the type conversion with a null address check.

• Recommendation: Consider changing the function or adding explicit conversion to the address type before 0 (**address**(0)).

4.3 BasicToken.sol Evaluation

4.3.1 **Medium Severity**

o This file contains the "*transfer*" function, which has the address as one of the input parameters, but does not check the address for a null value (in the Ethereum virtual machine, omitted values are interpreted as null values), which means that token loss is possible when using this function (they will be sent to the address 0x0).

• Recommendation: Implement a null address check.

4.3.2 **Low Severity**

o This file contains the "*onlyPayloadSize*" modifier, which protects against one of the known types of attacks. However, the use of this modifier is not rational in some cases.

• Recommendation: Consider performing this test at a higher abstraction level.

o This file contains the "*transfer*" function, which has the number of transmitted tokens as one of the input parameters, but it does not check the balance at the sender's address to verify that enough tokens are available. The level of severity is low, since this test occurs in the SafeMath library, but it's not apparent.

• Recommendation: Implement a balance check for the required number of tokens.

4.4 ERC20.sol and ERC20BASIC.sol Evaluation

These files contain interfaces for the ERC20 standard and do not require any changes.

4.5 StandardToken.sol Evaluation

4.5.1 **Medium Severity**

o This file contains the "*transferFrom*" function, which has the address as one of the input parameters, but there is no check for a null address, which means that token loss is possible when using this function.

• Recommendation: Implement a check for a null address.

4.5.2 **Low Severity**

o This file contains the "*transferFrom*" function, which has the number of transmitted tokens as one of the input parameters, but it does not check the balance at the sender's address to verify that enough tokens are available. The level of severity is low, since this test occurs in the SafeMath library, but it's not apparent. The same is true of the check for the number of "approved" tokens (tokens transmitted using the approve function).

• Recommendation: Implement a balance check and check "approved" tokens for the required number of tokens.

o This file contains the "*approve*" function, which has a check to deflect attack vectors for the Approve/TransferFrom functions. However, there are no functions to increase or decrease the amount of "approved" tokens, which leads to less flexibility in the use of tokens (the number of "approved" tokens must always be reduced to 0 first, which means there are two function calls instead of one).

• Recommendation: To reduce overhead costs, implement increaseApproval and decreaseApproval functions where this is appropriate.

4.6 MintableToken.sol Evaluation

No vulnerabilities were detected in this file.

5 General Comments

5.1 Medium Vulnerability Level

• There is no protection from transferring tokens to the address of the contract (this situation is quite common). And since the contract does not provide any functions for handling tokens on the contract's balance, any tokens accidentally transferred to the contract's address will be lost.

5.2 Low Vulnerability Level

• For consistency in calculations, the SafeMath library could be used everywhere instead of ++.

6 Conclusion

This contract has minor recommendations that may affect certain functions or create inconveniences when using the contract. However, they will not prevent the contract from functioning. Serious vulnerabilities were not detected.

7 Tools Used

The audit of the contract code was conducted using the Remix IDE:
http://remix.ethereum.org

The Solidity compiler version used was ***0.4.17*** (the most recent stable version).