CryptoSec

SECURITY FOR BLOCKCHAIN

REPORT
STORIQA TOKEN SMART CONTRACT AUDIT RESULTS FOR THE STORIQA

CryptoSec

SECURITY FOR BLOCKCHAIN

Change history

Version	Date	Author	Changes
1.0	22.12.2017	CryptoSec	Report created

Confirmation

Name	Company	Position	Location	Email

Contacts

Company name	Tomahawk Technologies Inc.
Address	1801 Wedemeyer, San Francisco, 94129
PGP	CCFD 364F 0180 287E 4DD6 A0AB 6CDF F9EC 1BAE D618
Email	info@cryptosec.us
Web	https://www.cryptosec.us

CryptoSec
SECURITY FOR BLOCKCHAIN

Table of Contents

1 Limitations on disclosure and usage of this report

This report has been developed by the company CryptoSec (the Service Provider) based on the result of an Internet resources (external resources) and company network (internal resources) security audit of the Storiqa company (the Client). The document contains information on vulnerabilities, their severity and methods of exploitation of those vulnerabilities, discovered in the process of the audit.

The information, presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Storiqa.

If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

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.

SOLC - A compiler for Solidity.

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 functioning in a number of scenarios, or creates a risk that the contract may be broken.

4 Main Results

4.1 Vulnerability table for all ABC contracts

Contract name	Vulnerability level			
	Critical	High	Medium	Low
ERC20Basic.sol	-	-	-	2
SafeMath.sol	-	-	-	1
BasicToken.sol	-	-	1	3
ERC20.sol	-	-	-	2
StandartToken.sol	-	-	1	3
CirculatingToken.sol	-	-	-	2
Multiowned.sol	-	-	2	11
MultiownedControlled.sol	-	-	1	2
MintableMultiownedToken.sol	-	-	1	6
STQToken.sol	-	-	-	2
Genaral comments	-	-	3	3
Totals	-	-	9	37

4.2 ERC20Basic.sol Evaluation

4.2.1 Low Severity

- ✗ Functions “balanceOf” and “transfer” have not a visibility specifier. Default visibility specifier is “public”.
- ✓ Recommendation: Add visibility specifier “public” for each function.

- ✗ The “constant” identifier has been deprecated, and now the “view” or “pure” identifier is used instead. However, the “constant” identifier is currently an alias of the “view” identifier. Also functions can be declared “view” in which case they promise not to modify the state.
- ✓ Recommendation: Replace the “constant” identifier with the “view” identifier for the “balanceOf” function.

4.3 SafeMath.sol

4.3.1 Low Severity

- ✗ The “constant” identifier has been deprecated, and now the “view” or “pure” identifier is used instead. However, the “constant” identifier is currently an alias of the “view” identifier. But functions can be declared “pure” in which case they promise not to read from or modify the state.
- ✓ Recommendation: Replace the “constant” identifier with the “pure” identifier for all functions.

4.4 BasicToken.sol

4.4.1 Medium Severity

- ✗ This file contains the "transfer" function, which has the address as one of the input parameters, but don’t 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.

4.4.2 Low Severity

- ✗ Functions “balanceOf” and “transfer” have not a visibility specifier. Default visibility specifier is “public”.
- ✓ Recommendation: Add visibility specifier “public” for each function.

- ✗ The “balances” map has not a visibility specifier. Default visibility specifier is “public”.
- ✓ Recommendation: Add visibility specifier “public” for this map.

- ✗ The “constant” identifier has been deprecated, and now the “view” or “pure” identifier is used instead. However, the “constant” identifier is currently an alias of the “view” identifier. Also functions can be declared “view” in which case they promise not to modify the state.
- ✓ Recommendation: Replace the “constant” identifier with the “view” identifier for the “balanceOf” function.

4.5 ERC20.sol

4.5.1 Low Severity

- ✗ Functions “allowance”, “approve” and “transferFrom” have not a visibility specifier. Default visibility specifier is “public”.
- ✓ Recommendation: Add visibility specifier “public” for each function.

- ✗ The “constant” identifier has been deprecated, and now the “view” or “pure” identifier is used instead. However, the “constant” identifier is currently an alias of the “view” identifier. Also functions can be declared “view” in which case they promise not to modify the state.
- ✓ Recommendation: Replace the “constant” identifier with the “view” identifier for the “allowance” function.

4.6 StandartToken.sol

4.6.1 Medium Severity

- ✗ This file contains the “transferFrom” function, which has the address as one of the input parameters, but don’t 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.

4.6.2 Low Severity

- ✗ Functions “allowance”, “approve” and “transferFrom” have not a visibility specifier. Default visibility specifier is “public”.
- ✓ Recommendation: Add visibility specifier “public” for each function.

- ✗ 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.

- ✗ The “constant” identifier has been deprecated, and now the “view” or “pure” identifier is used instead. However, the “constant” identifier is currently an alias of the “view” identifier. Also functions can be declared “view” in which case they promise not to modify the state.
- ✓ Recommendation: Replace the “constant” identifier with the “view” identifier for the “allowance” function.

4.7 CirculatingToken.sol

4.7.1 Low Severity

- ✗ Functions “transfer”, “approve” and “transferFrom” have not a visibility specifier. Default visibility specifier is “public”.
- ✓ Recommendation: Add visibility specifier “public” for each function.

- ✗ The "enableCirculation" function has control structures “if” for check of condition. However, the convenience function “assert” can be used to check for conditions and throw an exception if the condition is not met.
- ✓ Recommendation: Replace the "if" control structures with the “assert” function.

4.8 Multiowned.sol

4.8.1 Medium Severity

- ✗ The "changeOwner" function has the modifier "onlymanyowners", where the "confirmAndCheck" function is then called. If number of operations is 512 in a queue, all operations with "pending" state are canceled. Thus when contract use intensively, regularly all “pending” operations will be canceled, and you will have to initiate operations again.
- ✓ Recommendation: Analyze this problem, and find an acceptable solution if necessary.

- ✗ This file contains the "changeOwner" and “addOwner” functions, which have the address as one of the input parameters, but don’t 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.

4.8.2 Low Severity

- ✗ The “multiowned” function has not a visibility specifier. Default visibility specifier is “public”.
- ✓ Recommendation: Add visibility specifier “public” for this function.

- ✗ 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 for “changeOwner”, “addOwner”, “removeOwner” and “changeRequirement” functions.

- ✗ The “constant” identifier has been deprecated, and now the “view” or “pure” identifier is used instead. However, the “constant” identifier is currently an alias of the “view”

identifier. But functions can be declared “pure” in which case they promise not to read from or modify the state.

✓ Recommendation: Replace the “constant” identifier with the “pure” identifier for the “checkOwnerIndex“ function.

× The “constant” identifier has been deprecated, and now the “view” or “pure” identifier is used instead. However, the “constant” identifier is currently an alias of the “view” identifier. Also functions can be declared “view” in which case they promise not to modify the state.

✓ Recommendation: Replace the “constant” identifier with the “view” identifier for “makeOwnerBitmapBit”, “isOperationActive”, “amIOwner”, “getOwner”, “getOwners”, “isOwner” functions.

× The variable “c_maxOwners” has not a visibility specifier. Default visibility specifier is “public”.

✓ Recommendation: Add visibility specifier “public” for this variable.

× The "multiowned" constructor function 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)).

× The "assertOwnersAreConsistent" function checks the variable “mOwners[0]” (type 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)).

× Functions "changeOwner" and “removeOwner” have the modifier "onlymanyowners", where the "confirmAndCheck" function is then called. In case 512 operations in queue function “clearPending” is called twice.

✓ Recommendation: Analyze this problem, and find an acceptable solution if necessary.

× The "removeOwner" function assigns a null value for “mOwners[ownerIndex]” (address data type). 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)).

× The "reorganizeOwners" function has implicit type conversions. We recommend to do this explicitly.

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

- ✗ The "changeRequirement" function calls “clearPending” function, which cancels all operations with "pending" state. Thus you will have to initiate operations again.
- ✓ Recommendation: Analyze this problem, and find an acceptable solution if necessary.

4.9 MultiownedControlled.sol

4.9.1 Medium Severity

- ✗ This file contains the "setController" function, which has the address as one of the input parameters, but don’t 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.

4.9.2 Low Severity

- ✗ The “MultiownedControlled” function has not a visibility specifier. Default visibility specifier is “public”.
- ✓ Recommendation: Add visibility specifier “public” for this function.

- ✗ 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 for the “setController” function.

4.10 MintableMultiownedToken

4.10.1 Medium Severity

- ✗ This file contains the "mint” function, which has the address as one of the input parameters, but don’t 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.

4.10.2 Low Severity

- ✗ Functions “MintableMultiownedToken”, “transfer” and “transferFrom” have not a visibility specifier. Default visibility specifier is “public”.
- ✓ Recommendation: Add visibility specifier “public” for each function.

- ✗ This contract contains “now” (alias for “block.timestamp”) thus miners can perform some manipulation. In this case miner manipulation risk is really low. But you can get event “Emission” with identically time.

- ✓ Recommendation: Consider the potential risk and use “block.number” if necessary.
- × The “divedensPool” variable has not a visibility specifier. Default visibility specifier is “public”.
- ✓ Recommendation: Add visibility specifier for this variable.
- × The “m_lastAccountEmission” map has not a visibility specifier. Default visibility specifier is “public”.
- ✓ Recommendation: Add visibility specifier for this map.
- × The “constant” identifier has been deprecated, and now the “view” or “pure” identifier is used instead. However, the “constant” identifier is currently an alias of the “view” identifier. Also functions can be declared “view” in which case they promise not to modify the state.
- ✓ Recommendation: Replace the “constant” identifier with the “view” identifier for “calculateDividendsFor” and “getLastEmissionNum” functions.
- × This file contains the "payDividendsTo” function, which has the address as one of the input parameters, but don’t 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). But severity is low, because “calculateDividendsFor” function is called further and this function has check for this parameter (require gas).
- ✓ Recommendation: Implement a null address check.

4.11 STQToken

4.11.1 Low Severity

- × The “SQTToken” function has not a visibility specifier. Default visibility specifier is “public”.
- ✓ Recommendation: Add visibility specifier “public” for this function.
- × 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 for the “setController” function.

4.12 General Comments

4.12.1 Medium Severity

- × 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 balance, any tokens accidentally transferred to the contract address will be lost.

✓ Recommendation: Add protection function for this case.

✗ Current code is written for old versions of solc. Use latest version of Solidity.

✓ Recommendation: Use solc version 0.4.20.

✗ The contract has implicit fallback function. Thus contract don't receive ether. We recommend to do this explicitly.

✓ Recommendation: Implement fallback function explicit without a "payable" marker.

4.12.2 Low Severity

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

✓ Recommendation: Use SafeMath library for all arithmetic actions.

✗ Code quality of the project is low, which made auditing it hard.

✓ Recommendation: improve code quality. Even better, use a coding style from solidity doc.

✗ Commented code adds clutter for the reader and creates unnecessary confusion.

✓ Recommendation: Remove all commented functions.

5 Conclusion

Serious vulnerabilities were not detected. Also 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.

6 Tools Used

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

The Solidity compiler version used was 0.4.20 (the most recent stable version).