AUDIT REPORT FOR QURREX ON QRX TOKEN SMART CONTRACTS

Provided by Tailor Swift

Introduction

Vulnerability Level

Low severity – A vulnerability that does not have a significant impact on the use of the contract and is probably subjective.

Medium severity – A vulnerability that could affect the desired outcome of executing the contract in certain scenarios.

High severity – A vulnerability that affects the desired outcome when using a contract, or provides an 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 the risk that the contract may be broken.

Evaluation of SafeMath contract

Low Severity

- This contract has “safeDiv” function, but this function is not used in contracts. Recommendation: Remove “safeDiv” function to save “gas”.
- Function “safeMul” has convenience function “assert” with null value check for input parameter “a”, thus function “safeMul” throws an exception and does not return 0 in case “a == 0”. Recommendation: Add the “if” control structures with “return 0” and move this check to “if” condition.

Evaluation of ERC20 contract

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. Also, functions must have the “view” identifier, when they are supposed to not modify the state. Recommendation: Replace the “constant” identifier with the “view” identifier for the “allowance” and “balanceOf” functions.

Evaluation of StandardToken contract

Low Severity

- Maps “balances” and “allowed” do not have a visibility specifier. The default visibility specifier is “public”. Recommendation: Add the “public” visibility specifier for each map.

- This contract 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: Implement “increaseApproval” and “decreaseApproval” functions where this is appropriate to reduce overhead costs.

Evaluation of Ownable contract

Low Severity

- This file contains the "transferOwnership" 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 the “pendingOwner” address is incorrect when using this function (the “pendingOwner” variable will be assigned to the address 0x0). Recommendation: Implement the null address check.
- The "claimOwnership" function initializes the “pendingOwner” address for a null value, but it is not the address data type with a null value. Now, the implicit type conversion is being performed by

the compiler, but it is recommended to do this explicitly. Recommendation: Before 0, add an explicit conversion to the address type (address(0)).

Evaluation of MultiOwnable contract

Low Severity

- The "ownerMap" map does not have a visibility specifier. Default visibility specifier is "public". Recommendation: Add the "public" visibility specifier for this map.

Evaluation of Pausable contract

Low Severity

- The "pause" function does not have a modifier. This function can be called anytime, but this only makes sense if the "paused" variable is "false". Recommendation: Add the "ifNotPaused" modifier for the "pause" function.
- Contract initializes the "paused" variable for the "false" value, but it is recommended to do this explicitly. Recommendation: Add an explicit initialization to the variable (paused = false).

Evaluation of BsToken contract

Low Severity

- The contract initializes the "locked" variable for a "false" value, but it is recommended to do this explicitly. Recommendation: Add an explicit initialization to the variable (locked = false).

- The “Lock” event is not used. Recommendation: Remove this event or add this event to the constructor.
- The constructor has the address as one of the input parameters, but doesn’t check the address for a null value (in the Ethereum virtual machine, omitted values are interpreted as null values), which means that the “seller” is not correct after the contract deployment (the “seller” will be initialized with address 0x0). Recommendation: Implement the null address check.
- The "changeSeller" function checks the input address for a null value, but it is not the address data type with a null value. Now, the implicit type conversion is performed by the compiler, but it is recommended to do this explicitly. Recommendation: Before 0x0, add an explicit conversion to the address type (address(0)).
- The “sell” function has the “if” control structure with the “require” convenience function. The “require” Function is used incorrectly, because the “safeSub” function has the same check. Recommendation: Remove the “require” function.

Evaluation of BsTokensale contract

Low Severity

- The “ReceivedWeiEvent” event does not have indexed parameters. Up to three parameters can receive the “indexed” attribute , which causes the respective arguments to be searched for: It is possible to filter by specific values of “indexed” arguments in the user interface. Recommendation: Add the “indexed” keyword for parameters if necessary.

- This contract contains “now” (alias for “block.timestamp”), thus miners can perform some manipulation. In this case, miner manipulation risk is low. Recommendation: Consider the potential risk and use “block.number” if necessary.
- The constructor has the address as one of the input parameters, but doesn’t check the address for a null value (in the Ethereum virtual machine, omitted values are interpreted as null values), which means that the “token” is incorrect after the contract deployment (the “token” will be initialized with address 0x0). Recommendation: Implement the null address check.
- The “_doSellTokens” function returns the “uint256” value, but this value is not used in contracts. Recommendation: Update the“_doSellTokens” function and remove the return value.

Evaluation of BsTokenIssuer contract

Low Severity

- The “TxRejected” event does not have indexed parameters. Up to three parameters can receive the “indexed” attribute , which causes the respective arguments to be searched for: It is possible to filter by specific values of the “indexed” arguments in the user interface. Recommendation: Add the “indexed” keyword for parameters if necessary.

- The constructor and the “setNotifier” 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 the “token” and the “notifier” are incorrect after the contract deployment and call the “setNotifier” function (the “token” and the “notifier” will be initialized with address 0x0). Recommendation: Implement the null address check.
- The “_issueTokensForOneTx” and “rejectTx” functions use variables with the “var” data type, but the “var” keyword is deprecated. Recommendation: Use the booleans data type in such cases.
- The "_issueTokensForOneTx" function checks the input address for a null value, but it is not the address data type with a null value. Now, the implicit type conversion is performed by the compiler, but it is recommended to do this explicitly. Recommendation: Before 0, add an explicit conversion to the address type (address(0)).

Evaluation of TestTokenIssuer contract

This contract does not require changes.

Evaluation of ProdTokenIssuer contract

This contract does not require changes.

Evaluation of ProdToken contract

This contract does not require changes.

Evaluation of PublicPreSale contract

This contract does not require changes.

General Comments

Low Severity

- For consistency in calculations, the SafeMath library should be used everywhere instead of “*”, “/”, “+=” or “-=”. Recommendation: Use the SafeMath library for all arithmetic actions.
- Current code is written for old versions of solc. Use the latest version of Solidity. Recommendation: Use solc version 0.4.20.

Conclusion

Minor improvements can be made, but the contract will function as intended even without these changes. Recommendation: Add tests for all smart contracts.

Tools

The Solidity compiler version 0.4.20 (the latest stable version).
The audit of the contract code is performed using Remix IDE (http://remix.ethereum.org).