Vendor Approved with

Liquid by Quoine

QUOINE

QRYPTOS

Novoa Smart Contract Audit

Date: 6 July 2018

Shopin

Shopin

Audit Summary: The Shopin token has **passed** the smart contract audit. Taken with our audit of their github repositories (Machine Learning and AI) [link] we can assert that Shopin is in good status based on current knowledge of Solidity flaws, automated testing, linting, and line by line analysis.

Assessment: PASSED

Report: Based on my review, I believe there are no special security concerns with the contract launching on mainnet and these tokens trading. Details below:

Legend: ***Library/functionName/Call*** technical concept ***Core Tech Stack***

Logic:

1. ***bulkAssign*** is basically just calling transfer repeatedly. Since it uses the ***message sender***'s tokens, it has no security issues beyond that of a regular transfer.
2. The only function ***Ownable*** has is to allow calling ***bulkAssign***. Since ***bulkAssign*** itself doesn't have issues, neither does ***Ownable***.
3. Essentially that means the only way to compromise the tokens is by compromising an account holding the tokens, which is true for any smart contract.

Best Practices Assessment: Aside from the contract itself, the unit tests for the token have problems which would affect a different.

1. ***JavaScript Numbers*** are being used instead of ***BN***. I don't think they're causing any problems currently, but it's a bad practice.
    a. If the number of decimals is increased, the tests may become unreliable, or break.

2. ***Test assertions*** are done using rounding, which is also a bad practice.
    a. The rounding place is also used incorrectly, as it is specifying 10^28 digits of precision, which is clearly not intended.

3. ***Number.toPrecision*** is used strangely and unnecessarily. It's called with an argument of 28, which is more than ***JavaScript/Node.js*** can guarantee precision.

Best Practices Result: It works and it safely does what is intended. There are best practices that should be applied for different contracts (detailed below) but this is **perfectly functional code**.

Best Practices Technical Details:

1. ***JavaScript Numbers*** are being used instead of ***BN***. I don't think they're causing any problems currently, but it's a bad practice.
    a. If the number of decimals is increased, the tests may become unreliable, or break.
    b. All numbers should be given using either ***BN*** or ***BigNumber***.
    c. Typically ***BN*** is recommended for interop with ***Web3.js***, since that uses ***BN***.

2. ***Test assertions*** are done using rounding, which is also a bad practice.
    a. Rounding problems are fairly common and hard to detect in ***financial code***.
        i. You don't want small errors going unnoticed in case they turn into large errors.
        ii. This is part of why large-precision number libraries like ***BN*** and ***BigNumber*** are used.
    b. The rounding place is also used incorrectly, as it is specifying 10^28 digits of precision.
        i. That would mean it's rounding to 10 quintillion decimal places.
        ii. I'm assuming it was intended to just be 28, but even that has problems because ***JavaScript/Node.js*** cannot guarantee 28 points of precision without external libraries.
    c. ***Number.toPrecision*** is used unnecessarily.
        i. It's called with an argument of 28, which is more than ***JavaScript/Node.js*** can guarantee precision.

3. ***CSV Path*** The path that the test token amounts take from the ***CSV*** file to the actual smart contract is rather convoluted. Keep in mind that each step has the potential for causing errors, whether it's with precision, or some other bug.

    a. The string '123' is read from one of the test data ***CSV*** files
    b. It is converted to a ***BigNumber(123)***
    c. The ***BigNumber*** is converted to a ***JavaScript Number: 123***
    d. ***Number.parseFloat*** is called with the ***Number 123***. Because it's not a String..
        i. It's first converted to string using ***Number.toString()***, when we get '123'.
        ii. ***parseFloat*** then parses the string into the number 123.
        iii. ***toPrecision(28)*** gets called on the number, which turns it into a string with 28 points of precision, like this: '123.0000000000000000000000000'
        iv. That string eventually makes it to ***Web3***'s encoding to integer function, which calls ***Math.trunc***
        v. ***Math.trunc*** then drops all decimals, so it's a string '123', and then converts it into an integer, 123.
        vi. Finally, ***Web3*** converts it into a hexadecimal: 0x7B, and it's sent to the ***Ethereum*** node.

Function Call Chaos Testing [Attack Bot] (PASSED)

This is implemented using our CryptoChaos engine which simulates an active hacker or attacker testing every major function of a token for vulnerabilities and undesired effects. Each section details the actual method calls attempted, the results of those attempts, and the specific arguments, function calls and hashes used in testing.

This method of testing is programmatic testing for underflows, overflows, undesired behavior, and other nuance that may lead to compromised or bugged token smart contracts. Shopin passed every exhaustive test for function calls, unintended behavior diagnosis, and other human and machine derived testing methodologies.

This test was inspired in part by the Parity multi-sig “hack”. A single user playing with known contract methods created an issue by calling “suicide” as a test and destroyed hundreds of millions of dollars worth of value overnight. This was a simple mistake on the contract developer’s part. Shopin has no such simple mistakes and has been exhaustively tested. See below.

Deployment

Deployment of the test contract.
Pass - deployed contract

Transfer(0x0000000000000000000000000000000000000000,
0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c, 1000000000)

Basic transfer

Basic test of the transfer function.
Pass - transfer succeeded
function transfer(0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C, 1000000)
events
Transfer(0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c,
0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C, 1000000)

Transfer too much

Test attempting to transfer more tokens than the sender's balance.
Pass - transfer prevented

function transfer(0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C, 1000000001)

Transfer maximum

Test transfering the maximum value of uint256 for potential overflow issues.
Pass - transfer prevented

function transfer(0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C,
115792089237316195423570985008687907853269984665640564039457584007913129639935)

Transfer from, without approval

Test using transferFrom without being approved.
Pass - transfer prevented

function transferFrom(0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C,
0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB, 1)

Approve

Basic test of the approve function.
Pass - approved

function approve(0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C, 1000000001)
events
Approval(0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c,
0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C, 1000000001)

Transfer from, higher than balance

Test transfering an approved amount that's higher than the source account's balance.

Pass - transfer prevented

function transferFrom(0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c, 0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB, 1000000001)

Decrease Approval

Test decreasing an approval.
Pass - decreased

function decreaseApproval(0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C, 1000000000)
events
Approval(0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c, 0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C, 1)

Transfer from, too much after decrease approval

Test transferring more than approved amount after decreased approval.
Pass - transfer prevented
function transferFrom(0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c, 0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB, 2)

Transfer from, remaining balance

Pass - transfer succeeded

Transfer the from account's remaining approved balance and check that approval is now zero.

function transferFrom(0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c, 0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB, 1)
events
Transfer(0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c, 0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB, 1)

Increase approval

Test changing approval to a higher value.
Pass - changed

function increaseApproval(0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C, 2)
events
Approval(0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c, 0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C, 2)

Transfer from, too much after decrease approval

Test transferFrom that's below the previous approval amount, but above the new approval amount.
Pass - transfer prevented

function transferFrom(0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c, 0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB, 3)

Transfer from, remaining balance after increase

Transfer higher than the account's remaining approved balance after increase

Pass - transfer prevented

function transferFrom(0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c, 0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB, 2)
events
Transfer(0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c, 0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB, 2)

Basic bulk assign

Pass - transfers succeeded
function bulkAssign([0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C, 0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB], [1000000, 2000000])
events
Transfer(0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c, 0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C, 1000000)
Transfer(0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c, 0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB, 2000000)

Bulk assign more than balance

Pass - transfers reverted
function bulkAssign([0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C, 0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C], [500000000, 500000001])

Bulk assign max value

Pass - transfers reverted
function bulkAssign([0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C, 0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C], [1, 115792089237316195423570985008687907853269984665640564039457584007913129639935])

Bulk assign mismatched arrays

Pass - transfers reverted
function bulkAssign([0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C, 0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C], [1, 2, 3])

Transfer ownership

Pass - ownership changed
function transferOwnership(0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C)
events

OwnershipTransferred(0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c,
0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C)

Bulk assign after no longer owner

Pass - bulk assign denied
function bulkAssign([0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C,
0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C], [1, 2])

Bulk assign from new owner

Pass - transfers succeeded
function bulkAssign([0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB,
0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB], [1, 2])
events
Transfer(0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C,
0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB, 1)
Transfer(0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C,
0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB, 2)