SOMISH

Blockchain Labs

Smart Contract Audit Report

Jointer Token

July 23rd, 2020

Amended on July 27th, 2020

Index

Summary

Audit Report prepared by Somish Blockchain Labs for Jointer Token smart contracts.

Process and Delivery

Two (2) independent experts performed an unbiased and isolated audit of the code below. The debrief took place on July 23rd, 2020, and the final results are presented here.

Audited Files

The following contract(s) were covered during the audit:

- Auction.sol
- AuctionProxy.sol
- AuctionTagAlong.sol
- AuctionTagAlongProxy.sol
- AuctionLiquadity.sol
- LiquadityProxy.sol
- Migrations.sol
- CurrencyPrice.sol
- PriceTracker.sol
- AuctionProtection.sol
- ProtectionProxy.sol
- TokenVault.sol
- TokenVaultProxy.sol
- EtnToken.sol
- Exchangeable.sol
- MainToken.sol
- StandardToken.sol
- StockToken.sol
- TokenUtils.sol
- WhiteList.sol
- WhiteListProxy.sol

- AuctionRegistery.sol
- MultiSigGovernance.sol

Client Documents

The following document(s) were provided by the client for the purpose of the audit:

- Jointer_technical_design_v7.2.docx.
- Copy for auditing Jointer Whitepapaper v7.9.docx.

Notes

Audit was performed on commit **4cd454e589265d0a91c8506de611cfc997be3525** of jointer-token repository and pragma version ^0.5.9.

Intended Behavior

The Jointer Auction is an open-ended fund used to invest in real estate and provide liquidity to the reserve. The auction is intended to run daily on a continuous basis. On each day whitelisted investors can contribute to the fund and earn JNTR assets individually and as a group. Each day, contributors have the opportunity to invest as much or as little as they desire while still benefiting from the purchasing power of the whole group.

Bonus Structure

Setting goals for each day of the Auction allows Jointer to provide investors an extra incentive to outperform the previous day.
The **Group Bonus** allows everyone to benefit from a greater JNTR discount once the auction reaches above previous day contribution.
The **Individual Bonus** benefits large contributors to the round by offering a multiplier that incentivizes daily lead investors.
A daily contribution cap of 150% of yesterday's contribution is placed, So contribution of any day should not go beyond 150% of yesterday's and bonus amount is uncapped.

Distribution of funds

90% of the investment is intended to be transferred to downside protection. 10% of remaining investment is intended to be allocated to Jointers liquidity reserve powered by Bancor Protocol while 90% is to be transferred to real estate fund wallet.

Downside Protection

For all investors, 90% of all contributed amounts will be locked in a downside protection contract. This will lock funds as well as JNTR tokens. The other 10% will be given to users with 10% of the tokens. If an investor cancels the investment before 365 days of investment, funds will be returned to investors and locked JNTR tokens will be sent to the company wallet. If the investor cancels the protection, funds will be transferred to the company and JNTR tokens will be transferred to the investor.

Stacking

Stacking gives users the opportunity to earn additional income while tokens are on the stacking contract. Users who have locked tokens on 90% downside protection can transfer locked tokens to stacking. Jointer will mint additional 1% of the daily base supply + group bonus. The reward will be distributed daily (by the end of the auction) among all users on Stacking in pro rata to their token amount. The reward will be added to their Staking amount. User can stop the stacking at any time and get all the cumulative tokens they have.

Jointers Liquidity Reserve

10% of the amount allocated to reserve will be transferred to main reserve and remaining 90% will be transferred to side reserve.

JNTR Token

JNTR is a liquidity bridge designed as a transfer of value between JNTR/ETN, JNTR/STOCK, and assets on the Ethereum network.

JNTR/ETN

JNTR/ETN is ERC20 compliant token. JNTR/ETN are minted whenever tokens are purchased. JNTR/ETN can be purchased only with JNTR, similarly when JNTR/ETN tokens are burned user gets JNTR. Investors can redeem their JNTR/ETN at any time for JNTR or JNTR/STOCK. Jointer can buy back the JNTR/ETN at any time see replacing the JNTR/ETN with an equal value of JNTR or cryptocurrency.

JNTR/STOCK

JNTR/STOCK is also a ERC20 compliant token.JNTR/STOCK assets represent preferred stock in Jointer. Jointer will pre-mint the necessary amount of JNTR/STOCK. JNTR/STOCK will also be minted when investors purchase the JNTR/STOCK asset using JNTR. Similarly when JNTR/STOCK tokens are burned user gets JNTR. Jointer can buy back the JNTR/STOCK at any time see replacing the JNTR/STOCK with an equal value of JNTR or cryptocurrency.

Issues Found

Critical

1. Incorrect address passed to daily contribution check calculation

When a user contributes in an auction using function contributeWithTokens(), the contract checks contribution limit via function calculateFund(). The _token argument to calculateFund() function is passed as address(0) which means that even though the contribution is via token, the limit is checked with the price of ETH instead of the token. This means that an auction can surpass the max contribution limit or disallow further contributions despite the limit not being met.

Recommendation

Pass address of token received by user as a parameter instead of address(0).

Amended (July 27th 2020): Option to contribute in an auction with tokens is removed. The issue is no longer present in commit 4fe200fbb8b5ddcdd9f36df5b5c55e79be32d41e.

2. Exposed function allowing anyone to withdraw protection funds

The function trasferExtraToken() in AuctionProtection.sol has no access restrictions. This allows anyone to withdraw all stacked JNTR token from the contract.

Recommendation

We would recommend removing the function if it is not intended to be used.

Amended (July 27th 2020): Function was removed from the AuctionProtection. The issue is no longer present in commit 4fe200fbb8b5ddcdd9f36df5b5c55e79be32d41e.

Major

1. Failing Unit Test Cases

The system involves investors money in ETH/tokens and has complex functionalities as per the intended behavior highlighted in the documentation. The test cases provided are not holistic and are failing. It is highly recommended to write unit test cases especially for a complex system like this and achieve 100% line and branch coverage to minimise the possibility of issues that are substantially difficult to track manually.

Amended (July 27th 2020): Broken test cases were fixed, however the total code coverage is under 50%.

Minor

1. auctionSoldOut flag was never set

In **fundAdded()** function of **Auction.sol,** a boolean variable named auctionSoldOut was used to check if the auction was sold out, but the variable was never set to true. So when a user contributes to the auction past the contribution limit, instead of reverting, the transaction takes place with zero contribution value returning back the funds to the user, which in turn leads to the wastage of users gas.
Amended (July 27th 2020): Flag was removed and any extra funds contributed after the contribution limit will be sent back to the user.

Notes

1. Upgrade code to Solidity version > 0.6.0

Currently, the smart contract is written in Solidity version 0.5.9, a now known vulnerable version of Solidity. The contracts should be updated and their compiler version fixed
Recommendation
Refer to the Ethereum changelog on the following link: (https://github.com/ethereum/solidity/blob/develop/Changelog.md) for the most up to date Solidity version. Contract elements may need to be updated in order to work with the latest Solidity version.

2. Follow the Solidity style guide

Consider following the Solidity style guide. It is intended to provide coding conventions for writing solidity code.

Recommendation
You can find documentation for Solidity Style guide on the following link:
https://solidity.readthedocs.io/en/v0.6.0/style-guide.html. Use solhint tool to check for linting errors.

3. Code commenting

Consider adding comments to functions and variable declarations in the smart contracts as it gives clarity to the reader and reduces confusion for the developers as well.

4. Gas Optimizations

Consider using keywords like **internal** and **external** instead of **public,** it will reduce gas consumption.

Recommendation
Consider using **internal** for the function which is callable only within the same contract and using **external** for the function which is callable only from outside of contract. It will optimize the function call as well as gas required for deployment.

5. No restriction on groupBonusRatio and mainTokenRatio

There is no percentage limit for group bonus and main token ratio. Owner can set the percentages more than 100.
Amended (July 27th 2020): As per the response of Jointer's team, the groupBonusRatio and mainTokenRatio can be more than 100%.

6. Faulty functions in interfaces

a. The function tokenAuctionEndPrice() is not present in the Auction contract but it was declared in IAuction.sol.
b. The function depositeToken() in AuctionTagAlong, TokenVault contracts returns boolean value but in their corresponding interfaces IAuctionTagAlong and ITokenVault the return value was not present

Amended (July 27th 2020): Fixed the interfaces and the issue is no longer present in commit 4fe200fbb8b5ddcdd9f36df5b5c55e79be32d41e.

7. Function ignores return value of approve()

The function approveTransferFrom() in TokenTransfer.sol ignores the return value of approve() function.
Amended (July 27th 2020): Placed the approve() function in require condition and the issue is no longer present in commit 4fe200fbb8b5ddcdd9f36df5b5c55e79be32d41e.

8. Complex way of storing proxy addresses

It is observed that a separate proxy registry is maintained for each contract, which makes the contract code base unnecessarily complicated and difficult to go through . We would recommend maintaining a single registry for all proxies like AuctionRegistry.sol.

SOMISH
Blockchain Labs

9. Use more reliable sources of price feed

The currency prices are updated through the provableAPI's. Maintaining a price feed like this is subject to errors, in case of postponed calls,failure of scripts etc. Additionally, maintaining a price feed like this will be very costly more so because of the recent surge in gas prices. We would recommend using a more reliable way of fetching the currency price feeds like Chainlink or Uniswap V2.

Amended (July 27th 2020): Jointer team has consciously chosen Provable as the most suitable for their purposes - obtaining prices from various independent sources (exchanges) as the use of prices offered by the Chainlink or Uniswap limits them to only one source.

Closing Summary

Upon audit of the Jointer's smart contract, it was observed that the contracts contain critical, major and minor issues, along with several areas of notes.

We recommend that the critical, major, minor issues should be resolved. Resolving for the areas of notes are up to Jointer's discretion. The notes refer to improving the operations of the smart contract.

Amended (July 27th 2020): All critical/major and minor Issues have been amended / fixed by the Jointer's team as per the comments made above. For the current state of issues and their amendments, we recommend the reader to thoroughly review the report.

Disclaimer

Somish Solutions Limited