SOMISH

Blockchain Labs

Smart Contract Audit Report

FishnChips

November 29th, 2019

Index

Summary

Audit Report prepared by Somish Blockchain Labs for FishnChips smart contract.

Process and Delivery

Two (2) independent experts performed an unbiased and isolated audit of the code below. The debrief took place on November 28th, 2019, and the final results are presented here.

Audited Files

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

- **finalhourglass.js**.

Client Documents

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

- Github Repository available at https://github.com/MCLXI/FishnChips

Notes

Audit was performed on commit 7e067d3bca0df1677a2c7de243166444b9d90216.

Intended Behavior

Details for the SELF-COMPOUNDING Hourglass…Fish n' Chips are as follows:

- 0.75 IOST FISH Token Buy In (Feeless) for ambassadors
- Max. 40 Ambassadors
- FISH tokens can never be sold.
- Ambassador Buy-in: Min. 50k IOST - Max 375k IOST

Chips token:

- Token: CHIPS
- Start Price: 1 IOST
- Fee: 11% In & 11% Out
- Increment: 0.1 increase per 1 mil IOST

Chips Buy In (with Referrals):

- 6% Divs
- 3% Referral
- 1% Fry Bankroll
- 1% Devs

Chips Sell:

- 8% Divs
- 1% Fry Bankroll
- 1% Devs
- 1% Floor (50% B&B, 50% B&D)

Node Earnings: (Operator and Voting side)
100% Voter Rewards are reinvested through Buyback
**Fish n' Chips Voting rewards:

- 50% Buyback & Burn, hourglass reinvests in itself by buying CHIPS with the voter rewards (also raises the token price every single day), then burns the CHIPS it holds.
- 50% Buyback & Distribute CHIPS to FISH Holders

Order goes:
Ambassadors (No fee, stable .75 rate) -> FRY Investment Fund (current around 3 million IOST, pays fees+rain divs on ambassadors) -> General Public

Two tokens:
FISH:

- Permanently locked, Can't be sold
- Ambassadors get FISH
- Everyone has the option to convert their CHIPS to FISH and lock them
- You can swap CHIPS for FISH but not the other way around,
- You get IOST divs just like CHIPS does + CHIPS divs from 50% Buyback & Distribute

CHIPS:

- Can be sold whenever, not locked
- Price changes with every buy and sell
- Receive divs in IOST

Issues Found

Critical

No critical Issues Found.

Major

1. Hard-coded array

The array of **ambassadors** is hard-coded in **ambassadorBuyIn()** that too in local variable.

Recommendation
Consider storing **ambassadors** list in **init()** and use a global variable to store it.

Amended (Nov 30th 2019): Issue was fixed by FishnChips Team and is no longer present in commit 66cfd3caba4d950035400a6326bba32fe2599801.

2. Ambassadors can purchase more than 375k worth IOST

As per the readme, ambassadors can own min. 50k IOST - max 375k IOST worth Fish. However, the check implemented is for purchase at a single purchase and does not consider the existing tokens owned by an ambassador. Ambassadors can buy more than the expected limit by triggering more than 1 transaction.

Recommendation
Consider adding a check for the total balance.

Amended (Nov 30th 2019): Issue related to maximum purchase worth 375k IOST was fixed by FishnChips Team and is no longer present in commit 66cfd3caba4d950035400a6326bba32fe2599801. But the function ambassadorBuyIn() expects a minimum of 50k IOST. This means that even if the 1st transaction is of amount greater than 50k IOST, if the second transaction is less than 50k IOST, it would be rejected.

Developer's Note: Function is working as intended. Please do not buy in with any less than 50k, even if done through multiple TX's.

3. Missing feature - raises the token price every single day

As per the readme, there should be an increase in token price every single day. However, we couldn't find any logic that updates token price on a daily basis.

Amended (Nov 30th 2019): Feature was removed by FishnChips Team and is no longer present in readme of commit b137aacc1ff3a413d8012ba2bf8c7c22b4117f40.

Developer's Note: The "feature" is simply an observation, not to be coded in.

4. Fish and Chips tokens are not following the IRC20 standard.

Tokens in the present project are not following the **IRC20** standard. It is recommended that one should use **IRC20** while dealing with tokens for better results. We understand that Fish n Chips are non transferable but other functions like **balanceOf**, **issue**, **destroy** make sense. One could restrict transfer and implement the rest of the interface.

Recommendation
Consider Using standard IRC20 for tokens.

Developer's Note: IRC20 is unsuitable for the current purposes as transfer cannot be restricted. IRC21 is not widely supported on IOST. We chose to use a TRC-20 style contact-level implementation of the tokens. Excess token management functions adds unnecessary bulk and confusion to the code.

5. buyin() will throw when it will be called by _buyBack().

The function **buyin()** will throw when it will be called by **_buyback()** because of condition **!blockchain.requireAuth(account, 'active')**.

Recommendation
Consider bypassing the authentication check if called by **_buyBack()**.

Amended (Nov 30th 2019): Issue was fixed by FishnChips Team and is no longer present in commit ebc381d65cdd515395170f533910ab431f94a243.

6. No way to upgrade the contract in the future.

Since there is no **can_update()** function, non-upgradability of contract can be a threat to the system. There shall be no way to handle any bugs identified in the future. We recommend, there should be some way to upgrade the contract in case of some unforeseen problems with the present code.

Recommendation
Consider adding **can_update()** function with appropriate checks and make it callable via some governance mechanism.

Developer's Note: Immutability is a feature of this contract. can_update is purposefully removed.

7. Implementation does not match requirements.

According to requirement, chips cost should start with 1 iost with 11% of fees distributed among various stakeholders. This ideally means that a deposit of 10 iost should give users 8.9 chips. However, the implementation contains some quadratic equation which results in users getting less than 8.9 chips.

Developer's Note: The function is working as intended. The spec may have been unclear at an increment of .1 IOST per 1 million is to be incremented per chip, NOT per million. i.e., price moves up 0.0000001 IOST for each chip bought. The first chip will cost 1.0000001, the second will cost 1.0000002, etc. Therefore, the correct answer must be under 8.9. The contract returns ~8.89999 as the correct answer which follows the formula of partial sum through individual chip incrementation. Further reading :https://en.wikipedia.org/wiki/1_%2B_2_%2B_3_%2B_4_%2B_%E2%8B%AF

8. Adding/subtracting 1 unnecessarily while _solve() and _calcIOSTReceived() respectively.

We couldn't understand the significance of adding/subtracting 1 in _solve() and _calcIOSTReceived() respectively. It may lead to mint extra tokens as if a user calls ambassadorBuyIn() or buyin() 10 times and calls sell() and sell all tokens in one go.

Developer's Note: Adding and subtracting 1 follows simulation tests to generate the correct answer per the partial sums formula for calculating slippage. These answers have been cross-checked with the while/for loop equivalent and can be verified through simulations that all numbers are accurate on buy and sell, regardless of size.
i.e., Buying 1 CHIP 10 times does not, and will never give you more chips than buying 10 at a time.

Minor

1. Denial of Service.

As **chipsCurrSupply** is initially 0, the function **buyin()** will throw due to div/0 exception at **Line no. 520**. No user will be able to buy chips as **chipsCurrSupply** will always be 0. This may result in a deadlock.

Recommendation
Consider handling div/0 cases properly.

Amended (Nov 30th 2019): This is left on intention to disallow purchases before ambassador. However, we still feel that you should check div/0 and throw an error message that the system is waiting for an ambassador to buy in 1st.

2. fishCurrSupply will result in div/0.

As **fishCurrSupply** is initially 0, For **fishCurrSupply** = 0, **voterrewards()** and **sell()** will throw div/0 exception due to **Line no. 279**. No user will be able to use the above functions as **fishCurrSupply** will be 0. We understand the function could work fine if the function **chipstofish()** is triggered first. as it will update **fishCurrSupply** to non-zero value. But still, there is a possibility of a deadlock, especially with the existence of issue 1.

Recommendation
Consider handling div/0 cases properly.

Amended (Nov 30th 2019): This is left on intention to disallow purchases before ambassador. However, we still feel that you should check div/0 and throw an error message that the system is waiting for an ambassador to buy in 1st.

3. No check for valid integers

Few functions don't have a check to ensure whether an integer input is a valid integer or not.
For example:- See functions like **claimAirdrops() ,nodeUnvote(),buyin(),nodeVote()**.

Recommendation
Consider adding a compatibility check for integer inputs.

Developer's Note: The IOST blockchain will throw an error by default on invalid integer input.

4. Incorrect modification of state variable in _buyBack().

In function **_buyBack**(), there is check that if **!storage.has('fishCurrSupply'),** make **profitPerShare_fish** = 0. However, if **!storage.has('fishCurrSupply')** is true for non-zero **profitPerShare_fish,** it will still make **profitPerShare_fish** = 0 which is not intended.

Recommendation
Consider removing that modification.

Amended (Nov 30th 2019): Issue was fixed by FishnChips Team and is no longer present in commit 66cfd3caba4d950035400a6326bba32fe2599801.

5. No way to update accounts of MC and Justin.

There are 2 authorized accounts that are hard-coded in a few places. Since there is no way to transfer their administrative rights, it can be a threat to the system in case any of these accounts are hacked. We recommend, there should be some way to update these accounts in case of some unforeseen problems with the present account.

Recommendation
Consider storing authorized accounts in global variables and make some function to transfer administrative rights to other accounts.

Developer's Note: Functions such as transferOwnership() only have relevance on solidity-based blockchains. IOST allows accounts to change their private key. A function to change owner is less optimal than simply changing the PK of the authorized accounts. NOTE: No authorized functions may be called until Dec. 5th, 2:00pm PST.

6. System maintaining a separate variable to store IOST balance of the contract.

The system is maintaining a separate variable to keep track of IOST balance in contract rather than using standard function.

Recommendation

Consider using standard **IRC20** function to get a balance of IOST wherever needed.

Developer's Note: See previous note why IRC20 was not an optimal solution for the contract.

Notes

1. Follow proper linting

Few blocks of code are not correctly indented.

Recommendation
Consider working on improving indentation and proper coding/naming conventions for writing code. It will increase code readability.

2. Hard-coded variables

There are instances where values are hard-coded which is not a best practice. It is always better to store them in variables and make them configurable in case if need to change parameters in the future.
Line numbers: 26, 32, 47, 51, 61, 64, 80, 182, 184, 186, 188, 287, 297, 315, 333, 350, 431, 434, 454, 456, 458, 460, 478, 493, 540, 553, 560, 573, 574, 575, 583, 595, 596, 597, 605, 640, 692, 704

Recommendation
Consider maintaining variables and use those variables rather than hard-coding it. Also, consider making them configurable for future usage.

3. Misleading reasons for revert

The revert message says "Sorry, cannot vote more than 20% of liquid funds." but the check is for "after the vote, it should still hold 20% liquid".

Line numbers: 330.

4. The code can be reusable.

_increasePrice() and **_decreasePrice()** are having almost the same code with the difference that **_decreasePrice()**, price is reducing to some value. Similarly, in function **_referralFee() if** and **else if** block has exactly the same code without any change.
Recommendation
Consider reusing the code, it makes the code less complex and easier to understand.

Closing Summary

Upon audit of FishnChips's smart contract, it was observed that the contract contains major and minor issues, along with several areas of notes.

We recommend that major and minor issues should be resolved. Resolving the areas of notes is up to FishnChips's discretion. The notes refer to improving the operations of the smart contract.

Disclaimer

Somish Blockchain Labs's audit is not a security warranty, investment advice, or an endorsement of the FishnChips's platform or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Somish from legal and financial liability.

Somish Solutions Limited