SOMISH

Blockchain Labs

Smart Contract Audit Report

HiP Token Contract

November 13th, 2019

Index

Summary

Audit Report prepared by Somish Blockchain Labs for HiP Token Contract.

Process and Delivery

Two (2) independent experts performed an unbiased and isolated audit of the code below. The debrief took place on November 13th, 2019, and the final results are presented here.

Audited Files

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

- HipToken.sol
- SafeMathLib.sol

Client Documents

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

- White paper - https://docs.google.com/document/d/1sIlvF1Rr1btPGqNZxopA-dUa8b47nuOXWYgzi0SUcOg
- One pager - https://docs.google.com/presentation/d/1zXnnKGfr9jro6Kn09ls7OCA19qH4je7w61t0N0ZP5n8
- Slide Deck - https://docs.google.com/presentation/d/1_QD5msKZDdiHNO5-hFZgP_WXN0d2iOjMDiP0mXtQrWA

Audit was performed on the HiP Token Repository commit number 8192716bc36a6e71ea6a8a8e77599422f65b491e and pragma version 0.5.0

Intended Behavior

HiP Token is an ERC20 compliant token. As such, it implements both the standard token methods and the standard token events.
In addition to the basic ERC20 functionalities, HiP Token also has the feature of locking tokens. Locking can be done through token holders wallet, in which user should mention amount of tokens to lock and locking period. The locked tokens cannot be transferred or burnt.

After the lock period is completed, the user can unlock the tokens, after which they can be transferred and burnt.

Issues Found

Critical

No critical issues found.

Major

1. Emit transfer() event while allocating initial supply to bank

A transfer event must be emitted when tokens are initially minted as per ERC20 standards.
Follow EIP20 standard for more information :https://eips.ethereum.org/EIPS/eip-20

Amended (Nov 13th 2019): Issue was fixed by HiP Token Team and is no longer present in commit `8ce646f6b3e34d741e3167fd96b1de29983ff4d6`.

2. Transfer frozen tokens to contract for maintaining proper totalSupply

Ideally totalSupply of a token is sum of balances of all token holders. But as per the current code frozen tokens are deducted from user's wallet and stored in another structure which defies the above statement.
The frozen tokens should be transferred to token contract address along with storing in another structure and when tokens are unlocked they can be transferred back from contract to user's wallet. By implementing the above logic, the variable `numCoinsFrozen` can also be removed as we can get number of frozen tokens by checking token balance of contract.

Amended (Nov 13th 2019): Issue was fixed by HiP Token Team and is no longer present in commit `8ce646f6b3e34d741e3167fd96b1de29983ff4d6`.

Minor

1. Missing check for zero addresses

Any tokens issued to zero address are permanently locked and result in erroneous token supply. A check for zero addresses (0x00...000) should be added to the following functions:

a. function `transfer()`
b. function `transferFrom()`
c. `constructor()`

Amended (Nov 13th 2019): Issue was fixed by HiP Token Team and is no longer present in commit `8ce646f6b3e34d741e3167fd96b1de29983ff4d6`.

2. Use separate functions for updating allowance

Follow the standard ERC20 pattern for increasing and decreasing allowance amount by using separate functions, which resolves the frontrun attacks.
Example of a frontrun attack:
For instance, consider Bob has given allowance of 100 tokens to Alice, but after some time, Bob makes a transaction for updating allowance to 90. Now, if Alice frontruns Bob's transaction with more amount of gas and transfers 90 tokens, then the updated allowance will still be 90 tokens. Ideally, the allowance should have been set to 0 as Alice has already transferred 90 tokens.

To handle this situation, standard ERC20's decreaseAllowance function is used (check https://github.com/ethereum/EIPs/issues/738 for reference). When Bob uses this function to decrease allowance by 10 tokens, even if Alice front-runs the transaction and transfers 90 tokens, the allowance limit post Bob's transaction will be set to zero.

Amended (Nov 13th 2019): Issue was fixed by HiP Token Team and is no longer present in commit `8ce646f6b3e34d741e3167fd96b1de29983ff4d6`.

3. Emit `transfer()` event while burning tokens

Transfer events are used as a standard to track the movement of tokens. It is a good practice to emit the transfer event whenever a token transaction is done. So a `transfer()` event with recipient as null address(0x000...000) must be emitted whenever tokens are brunt.

Amended (Nov 13th 2019): Issue was fixed by HiP Token Team and is no longer present in commit 8ce646f6b3e34d741e3167fd96b1de29983ff4d6.

4. Zero check for time period and number of tokens to lock

It is recommended to add zero input check in freeze() which restricts locking of tokens for nothing and wastage of gas amount.

Amended (Nov 13th 2019): Zero input check was added for number of tokens by the HiP Token Team in the commit 8ce646f6b3e34d741e3167fd96b1de29983ff4d6. Would recommend a check for zero days to be added as well.

Notes

1. 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. Comments are missing in the contract.

Amended (Nov 13th 2019): Comments were added by the HiP Token Team in the commit 8ce646f6b3e34d741e3167fd96b1de29983ff4d6. However, it is recommended to follow the pattern given in official documentation of solidity.

2. Add reasons for assert()

Consider adding messages for all messages whenever contract throws. it gives clarity to the users.

Amended (Nov 13th 2019): Issue was fixed by HiP Token Team and is no longer present in commit 8ce646f6b3e34d741e3167fd96b1de29983ff4d6.

3. Add name and symbol for token

It is recommended to add name and symbol for token address which makes it easier to identify token.

Amended (Nov 13th 2019): Issue was fixed by HiP Token Team and is no longer present in commit 8ce646f6b3e34d741e3167fd96b1de29983ff4d6.

4. Gas Optimizations

Consider making variables like balances, frozenBalances **private** as these are accessible via separate getter functions. Additionally, using keywords like **internal** and **external** instead of **public** for functions also helps 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.

Amended (Nov 13th 2019): HiP Token team has made unnecessary public variables to private in the commit 8ce646f6b3e34d741e3167fd96b1de29983ff4d6.

5. 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/v0.5.7/style-guide.html. Use solhint tool to check for linting errors.

Closing Summary

Upon audit of the HiP Token smart contract, it was observed that the contracts contain major and minor issues, along with several areas of notes.

We recommend that the major and minor issues should be resolved. Resolving the areas of notes are up to HiP'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 HiP Token platform or its products. This audit does not provide a security or correctness guarantee of the audited smart contract.

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