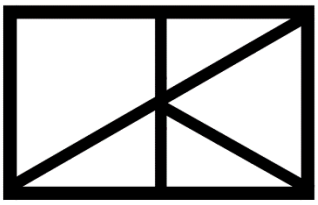
Lynchpin Smart Contracts Audit

Completed : 23 April 2019
Updated : 25 April 2020



Nakka

Prepared by Nakka Pte Ltd
mail@athng.com
http://www.nakka.sg

OVERVIEW

This document is a audit session for two smart contract of private ICO for Lynchpin. They are:

- Private ICO contract (general) - 0x2C924b8A92E293707142759c84cc7A8bED9c4834 :A private ICO contract for the general public which keeps tokens in escrow for one year after the start of crowdsale. Holders can withdraw tokens after 9 months by paying varying fee.Maximum tokens in this contract is 2 millions.

- Private ICO contract (team) - 0x28a3eaef0c37a787ace890c22ce2c2fafe8a3ae3 : A private ICO contract for the team which keeps tokens in escrow for 1 year after the crowdsale starts. Maximum tokens in this contract is 1 million.

SCOPE

This audit session is limited to 2 file smart contract for private ICO(described above). It including 'automation analysis' and 'manual code analysis'. Unit testing was performed, of which test cases are included as accompanying files.

EXECUTIVE SUMMARY

This Security Audit was initially performed in 23 April 2019. The rigorous assessment identified low severity issues that have since been fixed as of 11 September 2019. Specific fixes were recommended and implemented afterwards in the private ICO contract (general): 0x751e6d63049ecece85d9f392e8eef529cb746256 and the private ICO contract (team): 0x9B72FDbD2df126707FdcF3aF86AF17F7f36c0D7c

AUDIT RESULTS

Automation Analyst:

There are 2 warning with Etherscan with very low severity for ICO(general):

- https://etherscan.io/solcbuginfo?a=IncorrectEventSignatureInLibraries
- https://etherscan.io/solcbuginfo?a=ABIEncoderV2PackedStorage

There are 4 warning with Etherscan with very low severity for ICO(team):

- https://etherscan.io/solcbuginfo?a=IncorrectEventSignatureInLibraries
- https://etherscan.io/solcbuginfo?a=ABIEncoderV2PackedStorage
- ExpExponentCleanup (medium/high-severity)
- EventStructWrongData (very low-severity)

Manual Code Analyst:

With ICO(Private):

- Line 161: prefer putting 'visibility level' to 'public : sometimes, users want to check if the ICO is closed or not.
- Line 190: recommend to use SafeMath '**mul**' function to prevent 'number overflow' possibility
- Line 230: can replace uint256 to uint8 due to limitation of variable is less than 100

```
216    function closeSale() external onlyOwner
217    {
218        require (now > LOCK_PERIOD_START);
219        //lynT.transfer(msg.sender, lynT.balanceOf(address(this)));  //try remove this
220        owner.transfer(address(this).balance);
221        crowdsaleClosed = true;
222        emit LogSaleClosed();
223    }
```

- In line '219'(commented line) : if we don't comment this line, when we closeSale() and user wants to withdraw their token:

```
225    function withdrawMyTokens () external
226    {
227        require (crowdsaleClosed);
228        require (tokensOwed[msg.sender] > 0);
229        require (now > LOCK_PERIOD_9_MONTH);
230
231        uint256 penalty = 0;
232        if(now > LOCK_PERIOD_END)
233            penalty = 0;
234        else if(now > LOCK_PERIOD_11_MONTH)
235            penalty = 20;
236        else if(now > LOCK_PERIOD_10_MONTH)
237            penalty = 30;
238        else
239            penalty = 40;
240
241        uint256 tokenBought = tokensOwed[msg.sender];
242        uint256 toSend = tokenBought.sub(tokenBought.mul(penalty).div(100));
243        tokensOwed[msg.sender] = 0;
244        lynT.transfer(msg.sender, toSend);//transfer from this.ico ->msg.sender
245
246    }
```

Line 244 tries to transfer 'toSend' amount from ICO address to user's address. But line '219' transferred all balance of ICO to 'owner' => transaction reverted

With ICO(team):

- Line 161: prefer putting ‘visibility level’ to ‘public : sometimes, users want to check if the ICO is closed or not
- Line 187: recommend to use SafeMath ‘**mul**’ function to prevent ‘number overflow’ possibility

- Line 197: **function giveTokens(address _receiver, uint _tokens) public onlyOwner :** suggest to check ‘**_receiver’** to prevent null address:
    - `require(_receiver != address(0));`

CONCLUSION

The assessment identified low severity issues that have since been fixed. Specific fixes were recommended and implemented in the revised contracts afterwards in the private ICO contract (general) : 0x751e6d63049ecece85d9f392e8eef529cb746256 and the private ICO contract (team): 0x9B72FDbD2df126707FdcF3aF86AF17F7f36c0D7c

Overall, the code reviewed is of good quality, written with the awareness of smart contract development best practices.