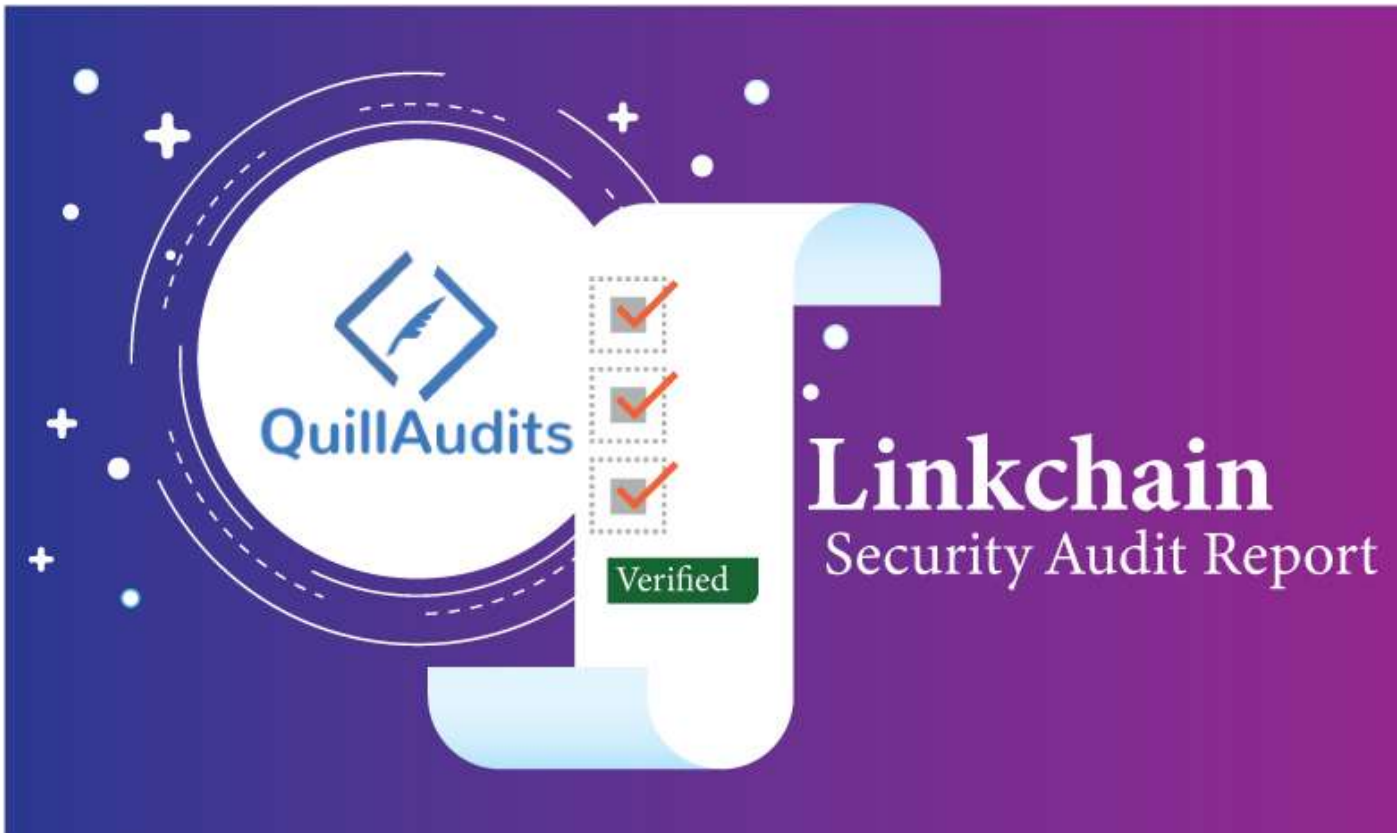
abhisharma1357 Update linkchain.md History

2 contributors

167 lines (93 sloc) | 7.3 KB

Linkchain smart contract audit report



> We've been asked by the Linkchain team to review and audit their smart contracts.

We at **QuillAudits** are a team of blockchain developers , consultants and security auditors , trying to build a more secure and safe community in the Blockchain space. QuillAudits, QuillHash Smart Contracts Security Audit platform ensures the reliability of your smart contract by complete assessment of your system's architecture and your smart contract codebase.

by QuillAudits Oct 3, 2018

The Code is located in the linkchain github Repository and the version used for this commit is

e09fe43816ff0d260f37664f1fda5298116a099e.

1. Introduction

This Audit Report highlights the overall security of LinkChain Smart Contracts. With this report, we have tried to ensure the reliability of their smart contract by complete assessment of their smart contract codebase.

1.1 Auditing Approach and Methodologies applied

Quillhash team has performed thorough testing of the project starting with analysing the code design patterns in which we reviewed the smart contract architecture to ensure it is structured and safe use of third party smart contracts and libraries.

Our team then performed a formal line by line inspection of the Smart Contract in order to find any potential issue like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.

In the **Unit testing** Phase we coded/conducted Custom unit tests written for each function in the contract to verify that each function works as expected. In Automated Testing, We tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

> The code was tested in collaboration of our multiple team members and this included :

```
Testing the functionality of the Smart Contract to determine proper logic has been
followed throughout.
Analyzing the complexity of the code by thorough, manual review of the code, line-by-
line.
Deploying the code on testnet using multiple clients to run live tests.
Analysing failure preparations to check how the Smart Contract performs in case of bugs
and vulnerabilities.
Checking whether all the libraries used in the code are on the latest version.
Analyzing the security of the on-chain data.
```

Security Level references

> Every issue in this report was assigned a severity level from the following:

High severity issues will probably bring problems and should be fixed

Medium severity issues could potentially bring problems and should eventually be fixed.

Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

High severity issues:-

1. No bounty function is present , but tokens for the same is reserved from ico token i.e BOUNTY_TOKENS = 2000000(declared,unused variable) wastage of tokens.
2. You have defined hardcap and softcap in terms of dollars in description and you are taking hard cap and softcap in ether in smart contracts.There must be some ether to dollar conversion mechanism by using oraclize or some manual updation using any function.
3. You are not multiplying totalSupply and other token pool with 10^{18} .As all wallets use last 18 digits as decimals for tokens .You must multiply totalSupply and other tokenPool like .

```
totalSupply = 700000000;

ADVISOR_TOKENS = 7350000

STRATEGIC_PARTNERS_TOKENS = 10000000

RESERVE_TOKENS = 109013333

OPERATIONS_TOKENS = 3150000

BOUNTY_TOKENS = 2000000

TEAM_TOKENS = 141820000

PRESALE_TOKENS = 56666667

PRIVATESALE_TOKENS = 100000000

CROWDSALE_TOKENS = 70000000

It should be :-

totalSupply = 700000000 * 1 ether;

ADVISOR_TOKENS = 7350000 * 1 ether

STRATEGIC_PARTNERS_TOKENS = 10000000 * 1 ether

RESERVE_TOKENS = 109013333 * 1 ether
```

```
OPERATIONS_TOKENS = 3150000 * 1 ether

BOUNTY_TOKENS = 2000000 * 1 ether

TEAM_TOKENS = 141820000 * 1 ether

PRESALE_TOKENS = 56666667 * 1 ether

PRIVATESALE_TOKENS = 100000000 * 1 ether

CROWDSALE_TOKENS = 70000000 * 1 ether
```

4. Minimum investment price is in terms of USD as provided in description, but in contract it is in terms of ether.

Medium Severity Issues:-

```
SafeMath library is not used in every mathematical function.

Some variables are declared but not used in program, opcodes are increased hence the gas

Declared, unused variables:

END_TIME_PRIVATESALE

START_TIME_PRESALE;

END_TIME_PRESALE;

2. You are using setHardCap() and setSoftCap() functions to change the hardcap and
softcap after contract deployment which is not a good thing as you are claiming refund
to investors if softcap is not reached.

3. “Stage can only increment one by one and has to be changed by owner.” This statement
is contradicted with the smart contract, you should check current stage and then
increment by one or use conditional statements.
```

Low Severity Issues

```
1. Declaration of a variables and function name is not standardised at all. It is very
important to use standard style define by solidity style-guide .

2. There is no reserve account as you mention in your deployment steps while deploying
constructor of crowdsale also start time end time parameters not present.

3. Function addAddressToWhitelist must have check the current stage of a sale before
whitelisting the user as stages can only be forwarded and can’t be reverse, so there is
no point to whitelist address who want to participate in already ended sale.

4. Solidity version must be fixed.
```

```
It should not pragma solidity ^0.4.24;

It should be pragma solidity 0.4.24;

5. According to standards ,only tokens for sale should be the parts of tokenSale
contract and rest of the functionality should be handled in token contract only.

6. You should use view keyword instead of constant in following functions:-

function allowance(address owner, address spender) constant public returns (uint256);

function balanceOf(address who) constant public returns (uint256);

function getStopReceive() constant public returns (bool);

7. Variable and function names should be meaningFul.Some of the variable names are
totally meaningless.

For ex: idtotokens does not make any sense.It should be like StageToTokens.If you are
calculating number of tokens in every stage of sale.
```

Comments:-

1. Constructor of crowdsale must check the argument with require statement, as it will be wastage of gas if wrong arguments is passed.
2. There is no conversion of ether to usd, as you mentioned conversion is dynamic.
3. Contract structure should be nice enough so that community can easily understand it.
4. Code should be properly commented.

Conclusion:-

These contracts are very unsafe to be used in production.You should review the suggested changes and implement them in smart contract.

You can request for Audit by filling a form : https://quillhash.typeform.com/to/KQ5Hhm

Thanks for reading. Also do check out our earlier blog posts. **QuillAudits** is a secure smart contract audits platform designed by QuillHash Technologies. It is a fully automated platform to verify smart contracts to check for security vulnerabilities through it's superior manual review and automated tools. We conduct both smart contract audits and penetration tests to find potential security vulnerabilities which might harm the platform's integrity.