PEXO SMART CONTRACTS AUDIT REPORT

by Quill Audits, January 2019

Introduction

This Audit Report highlights the overall security of Pexo Smart Contract. With this report, we have tried to ensure the reliability of their smart contract by complete assessment of their system's architecture and the smart contract codebase.

Auditing Approach and Methodologies applied -

QuillAudit 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 -

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

Summary of PEXO Smart Contract -

The Pexo Smart contract is functionally an ERC-20 Smart Contract. The smart contract has all the functions of an ERC20 standard with the addition of a function of Crowdsale, Pexo Smart contract aims to raise 70 mil USD(Hard cap).

Tokenomics :

Total Supply : 400 million

Tokens For Private Sale : 80 million (20%)

Tokens For Public Sale : 100 million (25%)

Tokens For Vesting (Team & Founders (5%), Advisors (5%), Rewards(15%), Retained(25%)) : 200 million (50%)

Tokens for Bounty (Bounty & marketing) : 20 million (5%)

Token Price : $ 0.25

Minimum Investment in Private Sale : $100

Discount in public Sale round one : 75 % , Token Price : $ 0.25

Discount in public Sale round Two : 40 % , Token Price : $ 0.60

Discount in public Sale round Three : 20 % , Token Price : $ 0.80

Discount in public Sale round Four : 10 % , Token Price : $ 0.90

Discount in public Sale round Five : 0 % , Token Price : $ 1

Contracts Descriptions based on Testing:

Contract PexoToken.sol

PexoToken.sol is a Token contract that hold 400 million (Total supply) tokens initially.

It contains all the ERC20 functionalities.

- activateSaleContract() this function activate sale contract and transfer 180 million tokens to crowdsale contract.
- activateVestingContract() this function allows vesting contract to transfer tokens that should go under vesting for 6 months from date of listing on exchange.
- Burn() this function allow to burn tokens .
- saleTransfer() only called by Sale contract and it will transfer tokens to beneficiary.
- vestingTransfer() only called by vesting contract and it will transfer tokens to beneficiary and will go under vesting for 6 months.
- burnTokensForSale() this function burn tokens that are unsold in Crowdsale Contract only after sale is over and called by owner only.
- freezeAccount() this function will stop activities of transfer, crowdsale participation of a address only called by owner.
- sendBounty() this will transfer tokens to any address, only called by owner.

Contract Crowdsale.sol

- authorizeKyc() this function will white list the address to participate in a crowdsale, only by owner and multiple address can be whitelisted in one transaction.
- pause() only by owner it will stop beneficiary to participate in a contract.
- restartSale() to restart sale when paused.
- startPrivateSale() to start private sale only by owner.
- endPrivateSale() to end private sale only by owner.
- startPublicSaleRoundOne() to start public sale round one only by owner.
- endPublicSaleRoundOne() to End public sale round one only by owner.
- startPublicSaleRoundTwo() to start public sale round Two only by owner.
- endPublicSaleRoundTwo() to End public sale round Two only by owner.
- startPublicSaleRoundThree() to start public sale round three only by owner.
- endPublicSaleRoundThree() to End public sale round three only by owner.
- startPublicSaleRoundFour() to start public sale round Four only by owner.
- endPublicSaleRoundFour() to End public sale round Four only by owner.
- startPublicSaleRoundFive() to start public sale round one only by owner.
- endPublicSaleRoundFive() to End public sale round one only by owner.
- getStage() will return current stage of crowdsale.
- sendPrivateSaleTokens() using this function you can send private sale tokens to beneficiary.
- setEthPriceInCents() to reset ether price in cents as already asked during deployment of crowdsale contract, only by owner .
- buyTokens() to buy tokens according to particular stage.

- ➢ discountInCurrentSale() this will return discount in particular stage.
- ➢ tokenAmount() this will return no of tokens you get in particular stage.
- ➢ burnTokens() this will burn all the token in crowdsale only after crowdsale is over, by owner only.
- ➢ finalizeSale() this will finalise the sale and after that only public investors will able to transfer tokens also token vesting will not available after finalising sale.
- ➢ SoftCapReached() this function will tell if softcap reached or not, soft cap is 700000000 usd in cents.
- ➢ tokenPriceInCurrentSale() this function will return token price in current sale.

Contract TokenVesting.sol

- ➢ vestTokens this function will vest tokens, called by owner only and vesting can be done before ico is over.

Unit Testing

Test Suite

Test cases:

Contract: PexoToken Contract, Crowdsale Contract, TokenVesting Contract

✓ Should correctly initialize constructor values of Pexo Token Contract (585ms)

✓ Should Deploy Crowdsale only (559ms)

✓ Should Deploy Vesting Contract only (165ms)

✓ Should set Vesting Contract Address to Pexo token Contract (369ms)

✓ Should Activate Sale contract (179ms)

✓ Should be able to pause Token contract (152ms)

✓ Should be able unPause Token contract (232ms)

✓ Should check balance of Crowdsale after, crowdsale activate from token contract (59ms)

✓ Should check balance of Vesting Contract, after Vesting activate from token contract (59ms)

✓ Should be able to check total tokens (96ms)

✓ Should be able to check tokens for sale

✓ Should be able to check tokens for Vesting contract

✓ Should be able to check tokens for Bounty (38ms)

✓ Should be able to check tokens for sale in crowdsale Contract (53ms)

✓ Should be able to check tokens for Private Sale (65ms)

✓ Should be able to check tokens for Public Sale (45ms)

✓ Should be able to check tokens for Vesting in Vesting contract (56ms)

✓ Should Authorize KYC for accounts [2], [3], [4] (210ms)

✓ Should Start CrowdSale (298ms)

✓ Should Send tokens to Private Investors (338ms)

✓ Should be able to pause Crowdsale contract (193ms)

✓ Should be able unPause Crowdsale contract (116ms)

✓ Should be able to freeze account (142ms)

✓ Should be able to Unfreeze account (129ms)

✓ Should End private sale (137ms)

✓ Should be able to vest tokens from vesting Contract (196ms)

✓ Should Start Public sale round one (167ms)

✓ Should be able to check Discount in Public sale round one

✓ Should be able to check Token price in Public sale round one

✓ Should be able to get correct token amount during public sale round one

✓ Should be able to buy Tokens according to Public sale Round One (2198ms)

✓ Should End Public sale round one and Start public Sale Round Two (218ms)

✓ Should be able to check Discount in Public sale round Two

✓ Should be able to check Token price in Public sale round Two

✓ Should be able to get correct token amount during public sale round Two

✓ Should be able to buy Tokens according to Public sale Round Two (2693ms)

✓ Should End Public sale round two and Start public Sale Round Three (468ms)

✓ Should be able to check Discount in Public sale round Three (72ms)

✓ Should be able to get correct token amount during public sale round Three (102ms)

✓ Should be able to check Token price in Public sale round Three (75ms)

✓ Should be able to buy Tokens according to Public sale Round Three (2521ms)

✓ Should End Public sale round Three and Start public Sale Round Four (285ms)

✓ Should be able to check Discount in Public sale round four

✓ Should be able to get correct token amount during public sale round Four (126ms)

✓ Should be able to check Token price in Public sale round Four (57ms)

✓ Should be able to buy Tokens according to Public sale Round Four (2494ms)

✓ Should End Public sale round four and Start public Sale Round five (238ms)

✓ Should be able to check Discount in Public sale round five

✓ Should be able to get correct token amount during public sale round Five

✓ Should be able to check Token price in Public sale round Five

✓ Should be able to buy Tokens according to Public sale Round Five (3056ms)

✓ Should End Public sale round five (178ms)

✓ Should be able to set ether price (130ms)

✓ Should be able to check if softCap reached (44ms)

✓ Should send Bounty Tokens (156ms)

✓ Should not be able to send Negative Bounty Tokens (40ms)

✓ Should be able to Burn Tokens (239ms)

✓ Should be able to Burn Crowdsale Contract Tokens After Sale is Over (147ms)

✓ Should be able to finalize Sale after sale is Over (304ms)

✓ Should be able to transfer Tokens got in public sale only (162ms)

✓ Should Not be able to transfer Tokens got in private sale (120ms)

✓ should Approve address to spend specific token (75ms)

✓ should increase Approval (94ms)

✓ Should be able to transfer Tokens on the behalf of accounts[4] (175ms)

✓ Should be able to transfer ownership of Crowdsale Contract (85ms)

✓ Should be able to Accept ownership of Crowdsale Contract

✓ should not increase Approval for Negative Tokens

Final Result of Test:

✓ 67 passing(23 s)

❌ 0 failing

Manual Testing(Rinkeby Test Network)

All the functions and variable are tested on test network (Rinkeby Test network), Input values and their output values are recorded with transaction hash.

https://docs.google.com/spreadsheets/d/1vaJgUwlCqMCK-0-0NCxE2HzS8l_UbFgKF97_0XFvnIU/edit?usp=sharing

Above sheet contain all the transaction hash, tested manually on rinkeby test net.

Test case, Test scenario, Test Data, Input Data, Expected Output, Actual Output, Status of test (pass/Fail) Transaction Hash.

CODE Coverage during unit Testing

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	93.49	59.73	90.41	91.25	
Crowdsale.sol	97.28	68.02	96.97	97.2	112,113,116,473
ERC20.sol	0	0	0	0	11,12
Owned.sol	100	50	100	100	
Pausable.sol	100	50	100	100	
PexoToken.sol	91.4	48.84	82.35	86.46	... 269,270,274
SafeMath.sol	100	62.5	100	100	
StandardToken.sol	74.19	43.75	85.71	73.08	... 114,115,117
TokenVesting.sol	100	50	66.67	100	
All files	93.49	59.73	90.41	91.25	

Issue Find in Contract :

According to a client specification document lock period for private investor and Team token(Vesting contract) are of six months after listed on exchange, but according to smart contract it is assumed that registration on exchange wll be done till ico completion and lock period will be started at the time of ico finalisation for next 6 months.

Solution :

This is subjective matter which need to be clear from client end whether lock period for private investor is start when ico is finalised or when listed on exchange or both will happen to be same ?