CYBER

www.cyberunit.tech

Kobizilla Smart Contract Audit

Date: 19th.Oct. 2021
Report for: Kobizilla
By: CyberUnit.Tech

This document may contain confidential information about IT systems and intellectual property of the customer as well as information about potential vulnerabilities and methods of their exploitation.
The report containing confidential information can be used internally by the customer, or it can disclose publicly after all vulnerabilities are fixed - upon the decision of the customer.

Scope and Code Revision Date

Initial Audit File	affiliate-platform-main.zip
Transaction SHA1Hash	0xf7e96eb78219948bb98749e0b3b4cffbc965b8f28b4be559c0e217153f21e25b
Initial Audit Date	9.09.2021
Secondary Audit.vl File	Kobizilla-contracts-main (2).zip
Secondary Audit_vlSHAl Hash	6391278f7bdf815654f79f44a4b4c1287cca33b
Secondary Audit vl Date	9.10.2021
Secondary Audit v2 File	Kobizilla-contracts-main.zip
Secondary Audit_v2 SHA1 Hash	c7c2f4e636bbe6c5677c38e5efO4c69a3f3124c
Secondary Audit v2 Date	9.11.2021
Secondary Audit v3 File	Kobizilla-contracts-develop.zip
Secondary Audit_v3 SHA1 Hash	48743ell96blc5324e581d9cl62db917e8a878f
Secondary Audit v3 Date	9.12.2021

Table of contents

Introduction

This report presents the findings of the security assessment of Customer's smart contract and its code review conducted between February 1st 2021 - February 7th 2021. Secondary audit was conducted between February 28th - March 2nd 2021; next remediation audit was conducted between April 1st - April 6th 2021; next remediation audit was conducted between April 8th - Oct.19th 2021.

Scope

The scope of the project is Kobizilla smart contracts, which was shared in the project folder:
Kobizilla-contracts-main.zip
Note. Only smart contracts in folder ico-contracts
We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the widely known vulnerabilities that considered (the full list includes them but is not limited to them):

Reentrancy
Timestamp Dependence
Gas Limit and Loops
DoS with (Unexpected) Throw
DoS with Block Gas Limit
Transaction-Ordering Dependence
Style guide violation
Transfer forwards all gas
BEP20 API violation
Compiler version not fixed
Unchecked external call - Unchecked math
Unsafe type inference
Implicit visibility level

Executive Summary

According to the assessment, Kobizilla smart contracts security risks are low.

Security engineers found several issues that were fixed before the final audit.

Our team performed an analysis of code functionality, manual audit and automated checks with Slither and remixed IDE. All issues found during automated investigation manually reviewed and application vulnerabilities presented in the Audit overview section. A general overview presented in the AS-IS section and all encountered matters can be found in the Audit overview section.

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss.
High	High-level vulnerabilities are difficult to exploit. However, they also have a significant impact on smart contract execution, e.g. public access to crucial functions.
Medium	Medium-level vulnerabilities are essential to fix; however, they can⁵t lead to tokens loss.
Low	Low-level vulnerabilities are mostly related to outdated or unused code snippets.
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can⁵t affect smart contract execution and can generally be ignored.

AS-IS overview

ICampaign

ICampaign is an interface for Campaign smart contract.

ICampaignFactory

ICampaignFactory is an interface for CampaignFactory smart contract.

IBEP20

IBEP20 is an interface for interactions with BEP20 tokens.

Initializable

Initializable is a standard OpenZeppelin contract for initializing contracts without constructors.

Ownable

Ownable is a standard OpenZeppelin contract for access control with an owner role.

Pausable

Pausable is a standard OpenZeppelin contract for pausing functions..

ReentrancyGuard

ReentrancyGuard is a standard OpenZeppelin contract that prevents reentrancy calls.

AggregatorV3Interface

AggregatorV3Interface is a smart contract for getting Ethereum price feed from ChainLink.

SafeMath

SafeMath is a standard OpenZeppelin library for math operations that prevents integer overflows.

Context

Context is a standard OpenZeppelin smart contract for providing execution context.

BEP20

BEP20 is a standard OpenZeppelin smart contract for BEP20 tokens.

NQT

NQT is a BEP20 custom implementation.

CampaignFactory

CampaignFactory is a smart contract for creating campaigns and managing fee parameters.

CampaignFactory is Ownable, Pausable, Initializable.

CampaignFactory has following parameters and structs:

```
int256 public platformFeeRate;
address public platformRevenueAddress;
address □ public allCampaigns;
mapping(address => mapping(address => address]])) public getCampaigns;
```

CampaignFactory contract has following functions:

- initialize - public functions that sets platformFeeRate and platformRevenueAddress
- getPlatformFeeRate - public view function that returns platform fee rate
- getplatformRevenueAddress - public view function that returns platform revenue address
- allCampaignsLength - public view function that returns total campaigns count
- getCreatedCampaignsByToken - public view function that returns campaigns for token and creator
- getCreatedCampaignsLengthByToken - public view function that returns campaign total amount for token and creator
- setPlatformFeeRate - external function for setting new platform fee rate. Has onlyOwner modifier
- setPlatformRevenueAddress - external function for setting a new revenue address. Has onlyOwner modifier
- registercampaign - external function for new campaign registering

Campaign

Campaign is a smart contract for campaign management.

Campaign is Ownable, ReentrancyGuard, Pausable.

Campaign has following parameters and structs:

```
BEP2O public token;
address public factory;
address public fundingWallet;
uint256 public openTime = now;
uint256 public closeTime;
uint256 public weiRaised = 0;
uint256 public tokenSold = 0;
uint256 public tokenClaimed = 0;
string public name;
uint256 private etherConversionRate;
uint256 private etherConversionRateDecimals = 0;
AggregatorV3Interface internal EthPriceFeed; mapping(address => uint256) private
tokenSoldMapping;
```

Campaign contract has following functions and modifiers:

tokenRateSet - modifier that checks whether token conversion rate is set constructor - public functions that sets deployer as campaign factory fallback - external function that always reverts receive - external payable function that calls buyTokenByEther initialize - external function that initializes contract variables getEtherConversionRate - public view function that returns ether conversion rate getEtherConversionRateDecimals - public view function that returns ether conversion rate decimals
getBuyableTokens - public view function that returns buyable tokens for address
getAvailableTokens - public view function that returns available to be sold
totalRaisedTokens - public view function that returns campaign token balance
getClaimableTokens - public view function that returns claimable amount for address
getLatestEthPrice - public view function that returns Ethereum price
setEtherConversionRate - public function that sets new ether conversion rate. Has onlyOwner modifier
setEtherConversionRateDecimals - public function that sets new ether conversion rate decimals. Has onlyOwner modifier
setChainlinkContract - public function that sets a new ChainLink address
setReleaseTime - public function that sets a new release time. Has onlyOwner modifier
setCloseTime - public function that sets a new close time. Has onlyOwner modifier
setOpenTime - public function that sets a new open time. Has onlyOwner modifier
buyTokenByEther - public payable function that performs token purchase by ether. Has nonReentrant modifier
isFinalized - public view function that returns true if ICO ended
isClaimable - public view function that returns true if ICO release time passed isOpen - public view function that returns true if ICO started refundTokenForIcoOwner - external function that returns unsold tokens to specified address. Has onlyOwner modifier
_preValidatePurchase - internal pure function that validates parameters for token

purchase

_getEtherToTokenAmount - internal view function that returns amount of tokens that can be purchased with specified amount of ether

_deliverTokens - internal function that performs ICO token transfer _forwardFunds - internal function that sends ether to funding wallet

_forwardTokenFunds - internal function that sends ERC2O token to funding wallet
_updatePurchasingState - internal function that updates wei raised and tokens sold amount

_updateDeliveryState - internal function that updates token balance for address
_validPurchase - internal view function that returns true if started and not ended
_payPlatformEtherFee - private function that transfer platform fee in ether to fee wallet

_getPlatformFeeRate - private view function that returns platform fee rate
_getPlatformRevenueAddress - private view function that returns platform revenue wallet

Audit overview

Critical

i. [Fixed] buyTokenByToken utilizes getLatestEthPrice as a USD cost of token, however, it returns Ethereum price. It may block users from buying an allowed amount of tokens.

Hig h

2. [Fixed] buyTokenByToken uses _updatePurchasingState to add total amount of weiRaised, however, it doesn5t raise Ethereum wei buy ERC2O token. Also, token decimals may be different from 18.
3. [Fixed] SafeMath library isn5t explicitly used for smart contracts. There are integer overflows in several places: in initialize function of Campaign (line 102); in _updatePurchasingState function of Campaign (lines 346,347).

Medium

4. [Fixed] NQT.sol file describes JST contract, however, contract name and file name should be descriptive.
5. [Fixed] It5s recommended to add PlatformFeeChanged event to CampaignFactory constructor, because it5s changed from 0.
6. [Fixed] It5s recommended to add events for revenue address change in CampaignFactory smart contract.
7. [Fixed] It5s impossible to set a higher token price than the price of a resource it will be bought for. For example, it5s impossible to set a token price to 2 ether.

Low

8. [Fixed] modifier tokenRateSet; isFinalized, isOpen, .transfer, _transferToken functions are never used within the smart contract. Consider removing them.
9. [Fixed] if releaseTime < closeTime there will be no buy limits (it can be potentially set up in constructor).
10. [Fixed] totalRaisedTokens doesn5t have a descriptive name - it doesn5t show the total raised tokens amount but maximum Campaign balance.
11. [Fixed] Solidity pragma version is not locked. It5s recommended to lock the Solidity version to the latest stable one.
12. [Fixed] _processPurchase function is an internal function that calls another internal function, thus, it can be removed.
13. [Fixed] Contract is not fully covered with functions documentation, it increases the risks of missed business logic issues.

Lowest / Code style / Best Practice

No vulnerabilities found.

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, high-level description of functionality presented in As-is overview section of the report.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security engineers found several vulnerabilities during the audits, however, all of them were fixed before final audit.

Disclaimer

The smart contracts given for audit have analyzed following the best industry practices at the date of this report, concerning: cybersecurity vulnerabilities and issues in smart contract source code, the details of which disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

The audit doesn't make warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the system, bug free status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is essential to note that you should not rely on this report only. We recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. Thus, the audit can^{5}t guarantee specific security of the audited smart contracts.