

**HACKEN**

# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

**Customer:** Embr Holdings Limited

**Date:** November 9<sup>th</sup>, 2021

This document may contain confidential information about IT systems and the 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 be disclosed publicly after all vulnerabilities are fixed – upon a decision of the Customer.

## Document

<b>Name</b>	Smart Contract Code Review and Security Analysis Report for Kindling Industries.
<b>Approved by</b>	Andrew Matiukhin   CTO Hacken OU
<b>Type</b>	BEP20 token; CrowdSale; Vault
<b>Platform</b>	Binance Smart Chain / Solidity
<b>Methods</b>	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
<b>Repository</b>	<a href="https://github.com/teamkindling/smart-contracts">https://github.com/teamkindling/smart-contracts</a>
<b>Commit</b>	bd830b5747421178227df0159fc5327b62f38c14
<b>Technical Documentation</b>	YES
<b>JS tests</b>	YES
<b>Website</b>	joinembr.com
<b>Timeline</b>	12 OCTOBER 2021 - 18 OCTOBER 2021
<b>Changelog</b>	18 OCTOBER 2021 - INITIAL AUDIT 01 NOVEMBER 2021 - SECOND REVIEW 09 NOVEMBER 2021 - THIRD REVIEW



## Table of contents

Introduction	4
Scope	4
Executive Summary	5
Severity Definitions	7
Audit overview	8
Conclusion	13
Disclaimers	14

## Introduction

Hacken OÜ (Consultant) was contracted by Kindling Industries (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contract and its code review conducted between October 12<sup>th</sup>, 2021 - October 18<sup>th</sup>, 2021.

Second review conducted on November 1<sup>st</sup>, 2021.

Third review conducted on November 9<sup>th</sup>, 2021.

## Scope

The scope of the project is smart contracts in the repository:

**Repository:**

<https://github.com/teamkindling/smart-contracts>

**Commit:**

[bd830b5747421178227df0159fc5327b62f38c14](#)

**Technical Documentation:** Yes (in repository [readme.md](#))

**JS tests:** Yes (in repository [test/](#))

**Contracts:**

[bep20\\_token.sol](#)

[crowdsale.sol](#)

[vault.sol](#)

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

Category	Check Item
Code review	<ul style="list-style-type: none"><li>▪ Reentrancy</li><li>▪ Ownership Takeover</li><li>▪ Timestamp Dependence</li><li>▪ Gas Limit and Loops</li><li>▪ DoS with (Unexpected) Throw</li><li>▪ DoS with Block Gas Limit</li><li>▪ Transaction-Ordering Dependence</li><li>▪ Style guide violation</li><li>▪ Costly Loop</li><li>▪ ERC20 API violation</li><li>▪ Unchecked external call</li><li>▪ Unchecked math</li><li>▪ Unsafe type inference</li><li>▪ Implicit visibility level</li><li>▪ Deployment Consistency</li><li>▪ Repository Consistency</li><li>▪ Data Consistency</li></ul>

Functional review	<ul style="list-style-type: none"> <li>▪ Business Logics Review</li> <li>▪ Functionality Checks</li> <li>▪ Access Control &amp; Authorization</li> <li>▪ Escrow manipulation</li> <li>▪ Token Supply manipulation</li> <li>▪ Assets integrity</li> <li>▪ User Balances manipulation</li> <li>▪ Data Consistency manipulation</li> <li>▪ Kill-Switch Mechanism</li> <li>▪ Operation Trails &amp; Event Generation</li> </ul>
-------------------	---

## Executive Summary

According to the assessment, the Customer's smart contracts are well-secured.



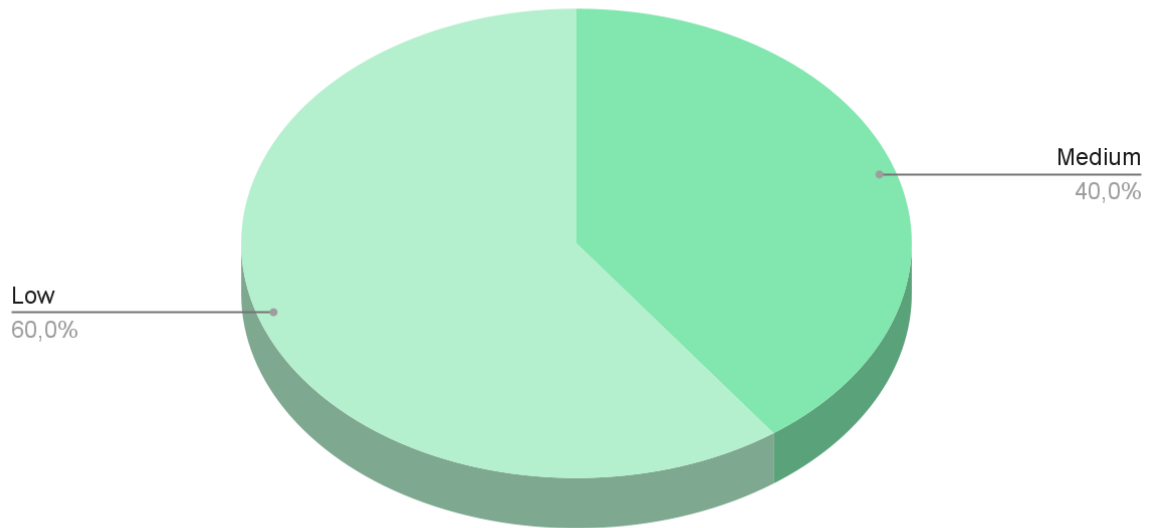
Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview section.

As a result of the audit, security engineers found **4** medium and **9** low severity issues.

After the second review and also considering comments added by the customer security engineers found that there are still unresolved **2** medium and **3** low severity issues.

After the third review security engineers found **1** low severity issue.

Graph 1. The distribution of vulnerabilities after the audit.



## Severity Definitions

Risk Level	Description
<b>Critical</b>	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
<b>High</b>	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
<b>Medium</b>	Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations.
<b>Low</b>	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution

## Audit overview

### ■ ■ ■ ■ Critical

No critical issues were found.

### ■ ■ ■ High

No high severity issues were found.

### ■ ■ Medium

1. Tests could not be run

While the documentation doesn't include an explanation on how to execute the tests, we've gone this way:

- truffle init
- npm i ethers
- truffle test test/test\_script.js

But, unfortunately, even when we match the solidity versions, no tests could be executed. Below is the only output of the script:

```
Compiling your contracts...
=====
> Everything is up to date, there is nothing to compile.

/* 1.deploy vault contract */

0 passing (0ms)
```

**Recommendation:** Please make sure all tests could be executed and there is a script or description of how to run them. Also, please make sure your tests are cover at least 95% of code branches.

**Status:** Fixed.

2. No emitting events

There are no emitting events neither in the CrowdSale nor the Vault contracts.

**Recommendation:** Please emit events on changing critical parameters as well as when performing contract actions.

**Status:** Responded.

**Customer comment:** *We have decided not to emit events as we don't require off-chain data in this case and to also reduce the gas used (up to 4x as advised by our Solidity consultant)*



### 3. No check for transfer result

While there could be any BEP20 token contract address set to the CrowdSale, not every contract will throw on error. BEP20 defines that a contract should return a boolean result of the transfer function, therefore there should be checking for the result.

**Contract:** crowdsale.sol

**Functions:** buy, getMyTokens

**Recommendation:** Please check the result of the transfer function call.

**Status:** Fixed.

### 4. No check for transfer result

While there could be any BEP20 token contract address set to the Vault, not every contract will throw on error. BEP20 defines that a contract should return a boolean result of the transfer function, therefore there should be checking for the result.

**Contract:** vault.sol

**Functions:** withdrawTokens

**Recommendation:** Please check the result of the transfer function call or use SafeTransfer library which is already doing the thing.

**Status:** Fixed.

## ■ Low

### 1. Tests configured incorrectly

We were able to run tests by the given instructions, but there are also some changes that should be made to accomplish that:

- rename "abi" => "abi-interfaces"
- line 17 of "test/test.js" change "abi-interfaces.vault.abi" => "abi-interfaces/vault.abi"

**Recommendation:** Please fix the test scripts.

**Status:** Fixed.

### 2. Tests running slow

As the docs stated: "The test may take over 45 minutes to run, due to dependency both on the public BSC testnet". But why not to fork the testnet and run tests in the local ganache environment with the ability to manually "mine" any number of blocks you need.

**Recommendation:** Please try to re-work tests to run them locally not remotely.

**Status:** Fixed.

### 3. Different solidity pragma versions in one codebase.



Using different solidity versions in one codebase make it harder to compile, deploy and test contracts.

**Recommendation:** Please use one Solidity version.

**Status:** Acknowledged.

**Customer Comment:** Leaving unchanged due to development constraints

4. Conformance to Solidity naming conventions

Solidity defines a [naming convention](#) that should be followed.

**Contract:** crowdsale.sol

**State Variables:** ADJ\_CONSTANT, \_own\_address, token\_address, wl\_script\_wallet, vault\_contract\_address, approved\_founders, approved\_founders\_length, sale\_stages, sale\_stages\_length, acive\_sale\_stage, withdraw\_schedules, withdraw\_schedules\_length, user\_schedule\_withdraw\_status, white\_list, user\_crowdsale\_balance, func\_sign, investors\_balances, init\_investors\_balances

**Function Parameters:** isApprovedFounder, setVaultContract, setActiveSaleStage, registerFounder, removeFounderByAddress, addSaleStage, addWithdrawSchedule, signFounder, resetSignatures, isAllFoundersSigned, emergencyWithdrawTokens, withdraw, hash, hash5uint256bool, getUserCrowdsaleBalance

**Recommendation:** Follow the Solidity [naming convention](#).

**Status:** Fixed.

5. Conformance to Solidity naming conventions

Solidity defines a [naming convention](#) that should be followed.

**Contract:** vault.sol

**State Variables:** \_own\_address, approved\_founders, approved\_founders\_length, allowance\_list, allowance\_list\_length, func\_sign

**Function Parameters:** isApprovedFounder, registerFounder, removeFounderByAddress, addAllowanceRecord, signFounder, resetSignatures, isAllFoundersSigned, isAllowedWithdrawal, getAllowedWithdrawal, withdrawTokens, emergencyWithdrawTokens, hash

**Recommendation:** Follow the Solidity [naming convention](#).

**Status:** Fixed.

6. No return statement

While the function declared to return a boolean value it will always return false without an explicit return statement.

**Contract:** crowdsale.sol

**Function:** resetSignatures



**Recommendation:** Please either use a return statement or remove the return type declaration.

**Status:** Fixed.

7. Dead-code

SafeMath library defines a method `mod(uint256,uint256)` that is never used in the code.

**Contract:** crowdsale.sol

**Functions:** SafeMath.mod

**Recommendation:** Remove unused functions.

**Status:** Fixed.

8. Too many digits

Literals with many digits are difficult to read and review.

**Contract:** crowdsale.sol

**Functions:** buy

**Constants:** ADJ\_CONSTANT

**Recommendation:** Please use scientific notation and/or ether units to simplify the numeric values (ie.: `100e3` instead of `100000`; `1 ether` instead of `1000000000000000000`)

**Status:** Partly Fixed. Constant is still contains many digits.

9. State variables that could be declared constant

Constant state variables should be declared constant to save gas.

**Contract:** crowdsale.sol

**State Variable:** ADJ\_CONSTANT

**Recommendation:** Add the `constant` attributes to state variables that never change.

**Status:** Fixed.

10. A public function that could be declared external

`public` functions that are never called by the contract should be declared `external` to save gas

**Contract:** bep20\_token.sol

**Functions:** name, symbol, decimals, totalSupply, balanceOf, transfer, allowance, approve, transferFrom, increaseAllowance, decreaseAllowance



**Recommendation:** Use the **external** attribute for functions never called from the contract.

**Status:** Fixed.

11. A public function that could be declared external

**public** functions that are never called by the contract should be declared **external** to save gas

**Contract:** crowdsale.sol

**Functions:** getActiveSaleStage, getActiveScheduleSlot

**Recommendation:** Use the **external** attribute for functions never called from the contract.

**Status:** Fixed.



## Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

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

As a result of the audit, security engineers found **4** medium and **9** low severity issues.

After the second review and also considering comments added by the customer security engineers found that there are still unresolved **2** medium and **3** low severity issues.

After the third review security engineers found **1** low severity issue.



## Disclaimers

### **Hacken Disclaimer**

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

The audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, 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 important 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 vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.