



Security Assessment

MCB Crowdsale

Apr 3rd, 2021



Summary

This report has been prepared for MCB Crowdsale smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Dynamic Analysis, Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in 8 findings that ranged from major to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	MCB Crowdsale
Description	a pre-purchase protocol
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/mcarloai/mcb-crowdsale
Commits	10eb9aa8da0c2f7f4815525012a9662fc6ca4f54

Audit Summary

Delivery Date	Apr 03, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Total Issues	9
● Critical	0
● Major	1
● Minor	3
● Informational	5
● Discussion	0

Audit Scope

ID	file	SHA256 Checksum
MCB	MCBCrowdsale.sol	a9dd2145fef795aa92a871b0e6019f0764d949ebaac2378c395b584a2aecadf0
MCV	MCBVesting.sol	e77e8bc02a583760c1f2b5b453b96a7300ce64e85fcca21c53b27f56dfc34d1c

Findings



Critical	0 (0.00%)
Major	1 (11.11%)
Minor	3 (33.33%)
Informational	5 (55.56%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
MCB-1	Proper Usage of "public" and "external" Type	Gas Optimization	● Informational	✓ Resolved
MCB-2	`Checks-effects-pattern` Not Used	Data Flow	● Minor	✓ Resolved
MCB-3	Assets Risk in `forwardFunds()`	Control Flow	● Major	✓ Resolved
MCB-4	State Variables That could be Declared Immutable	Gas Optimization	● Informational	ⓘ Acknowledged
MCB-5	File Not Found	Compiler Error	● Minor	✓ Resolved
MCV-1	File Not Found	Compiler Error	● Minor	✓ Resolved
MCV-2	Functions can be Restricted to Pure	Gas Optimization	● Informational	ⓘ Acknowledged
MCB-6	Functions can be Restricted to Pure	Gas Optimization	● Informational	ⓘ Acknowledged
MCV-3	Proper Usage of "public" and "external" Type	Gas Optimization	● Informational	✓ Resolved

MCB-1 | Proper Usage of "public" and "external" Type

Category	Severity	Location	Status
Gas Optimization	● Informational	MCBCrowdsale.sol: 60, 92, 113, 120, 129, 151, 175, 192	🟢 Resolved

Description

"public" functions that are never called by the contract could be declared "external". When the inputs are arrays "external" functions are more efficient than "public" functions. Examples: Functions `setEmergency()`, `totalSubscription()`, `subscriptionOf()`, `shareOf()`, `subscribe()`, `settle()`, `forwardFunds()`, `emergencySettle()`, `emergencyForwardFunds()` on the aforementioned lines.

Recommendation

Consider using the "external" attribute for functions never called from the contract.

Alleviation

The team heeded our advice and resolved this issue in commit `bf1029807da85d32ad3de17c1ccd62ace05ff3cb`.

MCB-2 | Checks-effects-pattern Not Used

Category	Severity	Location	Status
Data Flow	● Minor	MCBCrowdsale.sol: 167, 182, 202	✓ Resolved

Description

During `settle()`, `forwardFunds()` and `emergencySettle()` function calls state variables for balance are changed after transfers are done. This might lead to reentrancy issue. The order of external call/transfer and storage manipulation must follow checks-effects-interactions pattern.

Recommendation

It is recommended to follow checks-effects-interactions pattern for cases like this. It shields public functions from re-entrancy attacks. It's always a good practice to follow this pattern. `checks-effects-interactions` pattern also applies to ERC20 tokens as they can inform the recipient of a transfer in certain implementations.

Reference: <https://docs.soliditylang.org/en/develop/security-considerations.html?highlight=check-effects%23use-the-checks-effects-interactions-pattern>

Alleviation

The team heeded our advice and resolved this issue in commit `5e8b20caf3881567ed6b3753dee405deebf9c462` and `65f6e9b609df85ba0ac9de57302d3d00a3b029b5`.

MCB-3 | Assets Risk in `forwardFunds()`

Category	Severity	Location	Status
Control Flow	● Major	MCBCrowdsale.sol: 175, 210	✓ Resolved

Description

The functions `forwardFunds()` transfer all the user deposit USDC to address zero.

```
_usdcToken().safeTransfer(_mcdexFoundation(), fundUSDC);
```

To bridge the trust gap between administrator and users, administrator needs to express a sincere attitude with the consideration of the administrator team's anonymousness. The administrator has the responsibility to notify users with the following capability of the administrator:

- Administrators can transfer assets in this contract under unpredicted cases via `emergencyForwardFunds()` method.

The advantage of 'emergencyForwardFunds()' method in the protocol is that the administrator reserves the ability to rescue the assets in this contract under unexpected cases. It is also worthy of note the downside of 'forwardFunds' method, where the treasury in this contract can be migrated to any addresses.

Recommendation

To improve the trustworthiness of the project, any dynamic runtime changes on the protocol should be notified to clients. Any plan to call this 'emergencyForwardFunds' method is better to move to the execution queue of Timelock, and also emit events.

Alleviation

The team heeded our advice and resolved this issue in commit `3cf35672f069d51c66a25b4f0d10b984eaa3dc5c`.

- Set `MCDEX_FOUNDATION_ADDRESS` to the real account to receive forward funds;
- remove `emergencyForwardFunds()` method;
- add `onlyOwner` modifier to function `forwardFunds()`

MCB-4 | State Variables That could be Declared Immutable

Category	Severity	Location	Status
Gas Optimization	● Informational	MCBCrowdsale.sol: 29, 30, 31	ⓘ Acknowledged

Description

Below variables change only once, better to define them as immutable to avoid gas consumption.

```
uint256 public beginTime;  
uint256 public endTime;  
uint256 public unlockTime;
```

Recommendation

Add immutable attributes to state variables that only change once. We recommend to change the codes like below examples:

```
uint256 public immutable beginTime;  
uint256 public immutable endTime;  
uint256 public immutable unlockTime;
```

MCB-5 | File Not Found

Category	Severity	Location	Status
Compiler Error	● Minor	MCBCrowdsale.sol: 12	☑ Resolved

Description

File not found.

```
import "hardhat/console.sol";
```

Recommendation

Consider to fix the compiler error.

Alleviation

The team heeded our advice and resolved this issue in commit 557a623c2974da40b35674aa0e44c387ff47f653 and 474efdbd5b2bb5d4c1fe2720d32d89f39f55050e.

MCV-1 | File Not Found

Category	Severity	Location	Status
Compiler Error	● Minor	MCBVesting.sol: 9	☑ Resolved

Description

File not found.

```
import "hardhat/console.sol";
```

Recommendation

Consider to fix the compiler error.

Alleviation

The team heeded our advice and resolved this issue in commit 557a623c2974da40b35674aa0e44c387ff47f653 and 474efdbd5b2bb5d4c1fe2720d32d89f39f55050e.

MCV-2 | Functions can be Restricted to Pure

Category	Severity	Location	Status
Gas Optimization	● Informational	MCBVesting.sol: 91	📄 Acknowledged

Description

Function state mutability can be restricted to pure on the aforementioned lines.

Recommendation

Consider to replace the `view` to `pure`.

MCB-6 | Functions can be Restricted to Pure

Category	Severity	Location	Status
Gas Optimization	● Informational	MCBCrowdsale.sol: 221, 225, 229	① Acknowledged

Description

Function state mutability can be restricted to pure on the aforementioned lines.

Recommendation

Consider to replace the `view` to `pure`.

MCV-3 | Proper Usage of "public" and "external" Type

Category	Severity	Location	Status
Gas Optimization	● Informational	MCBVesting.sol: 57, 73	🟢 Resolved

Description

"public" functions that are never called by the contract could be declared "external". When the inputs are arrays "external" functions are more efficient than "public" functions. Examples: Functions `claimableToken()`, `claim()` on the aforementioned lines.

Recommendation

Consider using the "external" attribute for functions never called from the contract.

Alleviation

The team heeded our advice and resolved this issue in commit `5610109058ac8f2ca99994051420ef1fbf9b544e`.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in storage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of `private` or `delete` .

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

