SMART CONTRACT AUDIT REPORT

For

CLX (Order #FO1466620DE2)

Disclaimer: This Smart Contract Audit and Security vulnerability analysis was performed based on the best of our expertise and the use of third party tools. We do not assume any guarantee or warrantee of this analysis.

Manual Audit Result:

Code has passed **All** the security checks, outlined below:

---

(1) Overflows & Underflows

=> Client has implemented SafeMath library to mitigate this vulnerability.

(2) Visibility & Delegatecall

=> Also known as, The Parity Hack, which occurs while misuse of Delegatecall.

=> No such issues found and visibility also properly addressed.

(3) Reentrancy (TheDAO hack)

=> Use of “require” function mitigated this vulnerability.

(4) Forcing ether to a contract

=> Contract's balance has never been used as guard.

(5) Call to the Unknown (DoS with unexpected revert)

=> Usually found in smart contract for Ponzi scheme.

(6) Short Address Attack

=> Client side input validation must do to prevent Hacker to input invalid input data as like short Address.

Automated Tools Report

(1) https://tool.smartdec.net/scan/84c1482c6b1d467a83dbe7500e694fec

(1.1) DoS by external function call in require – **No Issues**

=> Tool provides 3 errors as not to relay on require function when calling from another contract. However, since it involves struct, this does not have any negative impact.

(1.2) Costly loop – **No Issues**

=> It gives 2 loops to check for the costly execution. But, since we have limited Phases, so the execution is not too lengthy.

(1.3) No payable fallback function – No Issues

=> Since this contract does not accept any ether, so it is ideal to revert all the ether back to sender.

(1.4) Reentrancy – **No Issues**

=> The tool trigger this error is due to involvement of struct and enum. But as mentioned in the tool website, it is a false positive. Tool also complains about transfer function because it is defined in interface.

(1.5) Timestamp dependence – **No Issues**

=> Because miners can slightly manipulate the timestamp, block's timestamp should be used for the critical component of the smart contract. By looking at the business model, the timestamp is mostly used to start/stop phases, so slight difference can be any issue.

(1.6) Unchecked math – **Not big issue**

=> Since safemath library is used in the smart contract, but it is not used in 3 places. However, by looking at the logic, it is not a big problematic.

(1.7) Overflow and Underflow in Solidity – **Not big Issue**

=> There is one occurrence of variable comparing with zero (Line: 408). But it is not a problem because it is guarded by an assert function.