XCELTRIP AUDIT

ARGUS

BLOCKCHAIN
AT BERKELEY

ARGUS AUDITS

Observe. Report. Defend.

AUTHORED BY:

ALI MOUSA
ALI@ARGUS.OBSERVER

PRANAV GADDAMADUGU
PRANAV@ARGUS.OBSERVER

COLLIN CHIN
COLLIN@ARGUS.OBSERVER

RAYMOND CHU
RAY@ARGUS.OBSERVER

XCELTRIP AUDIT

The Argus audit team performed a security audit of the Xcel Token Sale.

The review process comprised of individual inspections of the Xcel Token Sale contracts by members of the audit team, followed by a group review of the codebase, tests, documentation, and supporting information. Proprietary and open-source tools were employed to determine the safety and correctness of the implementation.

As of 3/10/2018 we have discovered several major bugs and lack of sufficient/correct testing. We recommend the team reevaluates the code and logic before deploying the contracts.

All of the discovered bugs have been rectified by the team as of the re-review of the code on 3/18/18. Code coverage has increased to up to 98% and tests generally cover most cases.

SCOPE

The team reviewed the code with the following metrics in mind:

Security

Identification of potential security flaws/attacks including but not limited to:

Data compliance
Integer overflow and underflow
Race conditions
DoS attacks
Timestamp dependence
Code quality

Reentrancy attacks
Sybil attacks
Front running
Transaction-ordering dependence
Call depth attacks

CODE QUALITY

Review of code to follow general best practices and ConsenSys Best Practices with a focus on:

Syntax
Readability
Consistency
Complexity
Scalability

Review of the smart contract architecture with primary focus on:

Game theory
Incentives
Pain Points
Usability

INITIAL FINDINGS

General Findings

SafeMath

Use SafeMath for all math related to the StepVesting contract to avoid rounding errors.

Require Statements

Use require statements instead of if statements and reverts.
StopVesting.sol: Lines 47-49
XcelToken.sol: Lines 121-123, 162-167, 179-181, 195-197, 212-217

Linter

We recommended linting your code for readability improvements.

Solidity Best Practices

We recommend reviewing the commonly accepted best practices when writing solidity code that explain how to avoid many of the bugs uncovered in this audit. The website is linked **here**

Testing

Line 93 in StepVesting.js test may be incorrect. (now - (this.start+this.cliffDuration)) / this.stepVestingDuration seems to be the intended behavior.

TestXcelToken.sol xcelToken.pause() doesn't work because emit keyword is not used for events in XcelToken.sol.

Test Coverage

Coverage conditions were identified using [solidity-coverage](https://github.com/sc-forks/solidity-coverage).

Lines 71-74 in XcelToken.sol modifier ```onlyTokenBuyer()``` is not tested.

Lines 77-80 in XcelToken.sol modifier ```nonZeroEth()``` is not tested.

Line 126 in XcelToken.sol ```require(isTeamVestingInitiated)``` branch condition is not tested.

Line 146 in XcelToken.sol ```require(loyaltyWallet != _loyaltyWallet)``` branch condition is not tested.

Line 162 in XcelToken.sol ```require(_totalWeiAmount > 0 && loyaltySupply >= _totalWeiAmount)``` multiple branch conditions are not tested.

Line 164 in XcelToken.sol ```require(transfer(loyaltyWallet, _totalWeiAmount) == true)``` branch condition is not tested.

Lines 217-219 in XcelToken.sol fallback function is not tested.

INITIAL FINDINGS

Significant Findings

Severity explanation:

High: Affects functionality of protocol
Medium: Affects intended functionality, but does not break protocol
Low: Design discrepancies, optimization opportunities, minor issues

High

Require Statements

Again, we recommend proper use of require statements. More can be read here

A note, It is sufficient, and in our opinion more clear, to write ```require(boolean expression)``` rather than ```require(boolean expression == true)```. XcelToken.sol: Lines 164, 208.

The fallback function of XcelToken.sol also should use ```revert();``` as opposed to ```require(false);``` as the latter is a rather unusual practice.

XcelToken.sol: Lines 103,104

There should be a require statement around transfer functions. XcelToken.sol(): Lines 131, 180, 194.

XcelToken.sol: initiateTeamVesting() Lines 126-127

Re-entrancy attack on lines 126-127. Mutex lock is set after the transfer.

XcelToken.sol: assignFoundationSupply() Lines 182-183

Re-entrancy attack on lines 182-183. Mutex lock is set after the transfer.

XcelToken.sol: assignReserveSupply() Lines 198-199

Re-entrancy attack on lines 198-199. Mutex lock is set after the transfer.

XcelToken.sol: allocateLoyaltySpend() Lines 162-166

Re-entrancy attack on lines 162-166. Subtracting balance after the transfer.

XcelToken.sol: buyTokens() Lines 212-216

Re-entrancy attack on lines 212-216. Subtracting balance after the transfer.

INITIAL FINDINGS

Medium

XcelToken.sol: setLoyaltyWallet() Lines 145-146

Should log events after successful change of wallet address.

XcelToken.sol: allocateLoyaltySpend() Line 160

Should use the `nonZeroAddress` modifier instead of the require statement.

Code Coverage, Oyente Reports, and SolGraph results can be found here

ARGUS
AUDITS

Contact
audit@argus.observer