SOLIDIFIED

Audit Report for VREO. June 29th, 2018.

Summary

Audit report prepared by Solidified, for Vreo covering the MERO token, crowdsale and KYC contracts.

Process and Delivery

Two (2) independent Solidified experts performed an unbiased and isolated audit of the below token sale. The debrief took place on June 26th, 2018 and the final results are presented here.

Audited Files

The following files were covered during the audit:

- VreoToken.sol
- VreoTokenBounty.sol
- PostKYCCrowdsale.sol
- VreoTokenSale.sol

Notes:
The audit was performed on commit `5e0b8c260c0a005bd3ed94ee8cf7c0951188e831`
github repo: http://github.com/Vreo/vreo-contracts
The audit was based on the solidity compiler `0.4.24+commit.e67f0147`

Intended Behavior

The purpose of these contracts is to operate a crowdsale for Vreo's platform token. The detailed specifications are available at:
https://github.com/Vreo/vreo-contracts/blob/master/docs/vreo-contract-specs.md

Issues Found

Critical

No critical vulnerabilities were identified.

Major

No major vulnerabilities were identified.

Minor

1. Assert used for user input validation

Because in some places SafeMath underflow protection is used to ensure that user submitted data produces valid contract state, in effect an assert is used to validate user input. This is not optimal because failed assert consumes all available gas in contrast with the best practice of validating user inputs using a require which refunds remaining gas to user. This problem is present in `VreoTokenSale.sol` on lines 116, 158 and 195 it's particularly severe in case of the `_deliverTokens` function because that is called during ordinary buy transactions and users nearing the token cap have no simple way of ensuring that their transaction won't cause the cap to be surpassed.

Recommendation

Add require checks before concerned safemath operations.

AMENDED [2018-06-27]

The issue has been fixed by the Vreo team and is no longer present in commit `a5b5fa567452573e684d277af792b829b9332245`.

2. User can't easily ensure that their buy transaction will be only included in the desired period

Since user can't specify the period with the associated bonus rate in their transactions and buy transactions for all periods have the identical format, user can't easily ensure that their transaction will be only included in the desired period. Their transaction can be delayed for different reasons from network congestion to malice on the part of miners or network middlemen.

Recommendation

We acknowledge that the likelihood of this problem being exploited is low and there are no elegant ways of integrating possible solution with the openzeppelin infrastructure. Ordinarily, the problem

would be solved by adding an optional argument to the buy function specifying sell period the buyer want to participate in.

AMENDED [2018-06-27]

VREO response

This is expected behavior. And and we feel adding a separate function to mitigate this tiny risk is not necessary.

3. Iconiq holders can transfer tokens to others during the sale, enabling users to buy more tokens than the defined amount set in the specs

---

Iconiq token holders can transfer their balance to other addresses, and by using multiple addresses they would be able to circumvent the investment limit set up in the contract/specs.

Recommendation

This issue could be mitigated by pausing the Iconiq contract during sale, or by using the balance in a specific block (would render impossible the balance check by the smart contract, the check could be made as part of the KYC process). A special ICONIQ token escrow contract holding user's balance during the sale could also be devised.

AMENDED [2018-06-27]

VREO response

We are aware of this. Since the ICONIQ token is not in our scope we accept people to buy more tokens buy transferring their ICONIQ token.

Notes

4. Checking that block time is after sale start and before sale end not strictly necessary in iconiqSaleOngoing & vreoSaleOngoing

Since sale time period check is already part of the underlying `TimedCrowdsale` contract `ICONIQ_SALE_OPENING_TIME <= now` on line 141 of `VreoTokenSale.sol` and `now <= VREO_SALE_CLOSING_TIME` on line 147 of `VreoTokenSale.sol` could be omitted without change in function.

AMENDED [2018-06-27]

VREO response

We think this check improves readability and it allows to use the the public view functions to see which sale in ongoing.

5. Arbitrary switching between `msg.sender` and `_beneficiary` in the span of one function

Line 92 of `PostKYCCrowdsale.sol` ensures that `msg.sender` and `_beneficiary` variables are identical, in the code these two are treated as such but switching between them in the span of one function seems a bit arbitrary and inconsistent. Concerned functions are: `PostKYCCrowdsale._processPurchase` and `VreoTokenSale._preValidatePurchase`.

AMENDED [2018-06-27]

The issue has been fixed by the Vreo team in `PostKYCCrowdsale` and is no longer present in commit `b81ce893e7606da037ace0c71f27bb43f2b440f9`. In addition the team added following explanation for why they've decided to maintain the use of both msg.sender and `_beneficiary` in `PostKYCCrowdsale._processPurchase`:

In "VreoTokenSale" Contract the method "_preValidatePurchase" continues using "_beneficiary" as a parameter, here is the reason:

1. This is the defined Open Zeppelin Interface

2. There is a need to pass it along to the method of the same name in the basic contract "PostKYCCrowdSale"

3. Only with entering "_preValidatePurchase" it is secure to consider "msg.sender" and "_beneficiary" as identical and from there on we use "msg.sender"

This consideration is a deliberate decision we made.

6. Note on manual Deployment

In the smart contract's specifications, the development team informs that the deployment will be performed manually, using remixD, Mist and a full Go Ethereum node. Consider performing an automated deployment with Truffle or similar tools. Automated deployment routines allow for testing, and mitigate risk of human error.

Closing Summary

VreoToken.sol has been verified as fully ERC20 compliant, and it contains the measures to mitigate the known EIP20 API Approve / TransferFrom multiple withdrawal attack.
Beyond the issues mentioned, the contracts were also checked for overflow/underflow issues, DoS and re-entrancy vulnerabilities. None were discovered.
OpenZeppelin contracts such as Ownable/SafeMath/Crowdsale/etc. have been widely audited and secured, as such, they were not prioritized for auditing.

Disclaimer

Audit Report for VREO. June 29th, 2018.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect auditors of Solidified platform from legal and financial liability.