SOLIDIFIED

Audit Report for Restart Energy ICO. January 13, 2018.

Summary

Audit Report prepared by Solidified for Restart Energy covering the ICO smart contracts referenced here.

Process and Delivery

Two (2) independent Solidified experts performed an unbiased and isolated audit of the below ICO contracts. The debrief and cross-verification took place on January 13, 2018 and the final results are presented here.

Audited Files

The following files were covered during the audit:

* **[RestartEnergyCrowdsale.sol]** (./contracts/RestartEnergyCrowdsale.sol)
* **[RestartEnergyToken.sol]** (./contracts/RestartEnergyToken.sol)
* **[TimedCrowdsale.sol]** (./contracts/TimedCrowdsale.sol)
* **[TokenCappedCrowdsale.sol]** (./contracts/TokenCappedCrowdsale.sol)
* **[Whitelistable.sol]** (./contracts/Whitelistable.sol)

Intended Behavior

The purpose of the contracts is to operate a crowdsale, distribute, and manage simple vesting of Restart's platform token.

A total of 500 000 000 tokens
TOKEN DISTRIBUTION:

80% for sale
20% for team/advisors/bounty (12% + 5% + 3%)
HardCap - 400 000 000 tokens

1 ETHER = 10 000 TOKENS (FOR THE BEGINNING)
Presale (from presaleStartTime to presaleEndTime)
Presale:
minimum 10 eth
20% bonus tokens
Sale (from startTime to endTime)
Bonus 15% during first 24 hours
Each day the bonus will decrease by 1% till 0%.

Token transfer is blocked until 15 days after Crowdsale end
Crowdsale end
14% of total Tokens are minted into team wallet
The bounty tokens are minted (3% of total Tokens)
The advisors tokens are minted (5% of total Tokens)
Half of the team's tokens are minted (50% from 12% of total Tokens => 6% from total Tokens)
The team can claim
25% from team's tokens only after 6 months after Crowdsale end (3% of total Tokens)
25% from team's tokens only after 12 months after Crowdsale end (3% of total Tokens)

There are two wallets involved
field wallet (for the the ETH from ICO)
field tokensWallet (for company/team/advisors/bounty/remaining Tokens)

Solidified Stamp

Maintaining the Solidified Stamp for Restart Energy requires addressing of issues (1-6) in the final version of the contracts and riding on confirmation of intended behaviour by the author.

Issues Found

1. Tokens are minted to the wrong wallet

In the function claimTeamTokens(), the tokens are being sent to wallet, instead of tokenWallet, as it should be, according to the intended behaviour. Critical status of this issue is dependent on the reasoning behind separating the eth and token wallets.
if the split was because the wallet for ether use is not ERC20 token compatible
then critical rating is justified.

Recommendation:

RestartEnergyCrowdsale.sol / lines 121,126

```
token.mint(wallet, tokensToMint);
```

Change `wallet` to `tokenWallet` in the claimTeamTokens function.

AMENDED [2018-1-14]:
This issue has been fixed by the Restart Energy team and is not present in `commit a89c78e`.

2. Consider that immediately minting the unsold tokens to `tokensWallet` on finalization can significantly alter the documented token allocations

---

Basing the percentage allocations on the amount of tokens being offered for sale rather than the amount of tokens sold is problematic. If more than 3-6% of tokens remain unsold, this stands to make the vesting period for team tokens appear insignificant. Additionally, the wider cryptocurrency community may view this as a way to manipulate the market cap of the token.

Recommendation:

This could be a deal breaker for investors. Ultimately, this is a business decision. Given time constraints, a change in the code is not recommended. However the team should include this behavior in the whitepaper and be transparent about the decision made, prepared to answer questions regarding this token distribution structure to the community/investors.

AMENDED [2018-1-14]:
This issue has been addressed by the Restart Energy team: any unsold tokens will be effectively burned. This functionality is present in `commit a89c78e`.

3. Tokens distributed through `sendTokensToAddress` are neither counted toward `soldTokens` nor logged with event

While not a security concern, tokens sold this way will not appear in the `soldTokens` metric. Thereby underreporting the success of the crowdsale. Additionally, the distributions that occur this way are not using the TokenPurchase event.

Recommendation:

RestartEnergyCrowdsale.sol / line 95

Add `soldTokens = soldTokens.add(tokens);` & `TokenPurchase(msg.sender, to, 0, amount);` to `sendTokensToAddress` function.

AMENDED [2018-1-14]:
This issue has been fixed by the Restart Energy team and is not present in `commit a89c78e`.

4. Presale is not enforced to happen before Crowdsale

There are no checks against the timeline of the presale event, making it possible to happen at the same time as the crowdsale, or not happen at all, if set to start and end in the past.

Recommendation:

Enforce the presaleStartTime and presaleEndTime are both in the future and before startTime. This could be done in the following way:

```
require(now <= presaleStartTime);
require(presaleEndTime > presaleStartTime);
require(presaleEndTime < startTime);
```

AMENDED [2018-1-14]:
This issue has been fixed by the Restart Energy team and is not present in `commit a89c78e`.

5. Whitelist functionality is included in the repo, but not implemented in the crowdsale

The repository includes [Whitelistable.sol] (./contracts/Whitelistable.sol) which refers to the presale as intended to be limited to whitelisted investors; however, the crowdsale neither implements nor enforces a whitelist. This is a problem both because of unnecessary files being deployed and because the possibility of misleading users and buyers that look to the deployed set of contracts.

Recommendation:

If the presale is intended to require whitelisting, RestartEnergyCrowdsale needs to inherit from Whitelistable, and check that `msg.sender` is on the whitelist needs to occur in `validPresalePurchase()`.

Otherwise remove Whitelistable.sol from the repo.

AMENDED [2018-1-14]:
This issue has been addressed by the Restart Energy team: they have removed the unused code and will not be using a whitelist for the crowdsale.

6. A Note Regarding the Bug Report Filed on Solidified Bounty

Solidified user "monty" has filed a bug report during the bounty program referenced here and stating:

> "Presale (from presaleStartTime to presaleEndTime)

> Sale (from startTime to endTime)"
>
> Meaning that the Presale ends before the beginning of the Sale, which implies:
> presaleEndTime <= startTime
>
> validPurchase() checks than:
> now >= startTime
>
> As a result, the line 68 will fail for a Presale if "presaleEndTime < startTime".
> In this case, no one can buy a token during the presale.

Both experts agree that this report is valid. The contract as written is clear in its intent for the presale and main sale periods to not overlap. In its current state it is not possible to purchase tokens during presale. The issue happens because in the buyTokens function, validPurchased() is called twice, and it throws because it checks against the timeframe of the token sale, not the presale.

Recommendation:

RestartEnergyCrowdsale.sol / line 70

Simply remove this line: `require(validPurchase());`

AMENDED [2018-1-14]:
This issue has been fixed by the Restart Energy team and is not present in commit a89c78e.

7. Sale does not finalize when tokens are sold out

If the hard cap is reached, the finalization function can not be triggered and the sale will remain active until the endTime arrives. Making all transactions in the meantime fail.

Recommendation:

Make the function hasEnded() (in Zeppelin's Crowdsale.sol) return true if cap is reached (tokenSold == hardcap)

AMENDED [2018-1-14]:
This issue has been addressed by the Restart Energy team: in the event of the crowdsale selling out, they will manually move the end date up (through function `setEndTime`) and finalize the contract.

8. Owner can modify sale conditions at any time

At any point during the sale, the owner can modify the rate, and the start and end time of both presale and crowdsale. This is a threat to the buyer security and to transparency, as none of the sale conditions stated in the whitepaper are enforceable and can change at any time.

Recommendation:

Limit the owner power, removing the ability to change such parameters. If, in any case, the sale needs to be postponed, deploy a new set of contracts with new, unmodifiable rules.

9. Purchase which would go over the sales hard cap is rejected, instead of fulfilled and refunded the excess ether

An investor attempting to purchase the last of the tokens available for sale will only be able to if their transaction is for the exact amount of tokens left-- no more. In this scenario it's often expected that the investor would instead receive as many tokens as possible and a refund of the unspent ether.

Recommendation:

Given the current time constraints, the potential attack surface opened up by implementing the ability to refund over-purchase is not worth the gain in functionality.

The team should be aware that this can happen, and prepare a response for investors in the unlikely event this does occur.

AMENDED [2018-1-14]:
This issue has been addressed by the Restart Energy team: they will be monitoring the crowdsale, and inform buyers of this limitation.

Additional Suggestions

Automatic Sale ending

The finalize function must be called from an external source, the contract itself does not detect that the sale has ended. If the owner never calls this function the finalization logic is never applied, and, therefore, the token is never transferable.

Recommendation:
Include finalization logic inside the buy function. It checks either if the endTime or the hardcap have been reached and triggers finalize() automatically

Misleading named variables

Some variables in the contracts have a bit of misleading name in relation to their behavior. Those are:

basicPresaleRate **and** etherRate
Readers infer that those two variables represent the amount of token units buyers get for spended ether, when in reality they are multiplied inside getRate(), modifying this value. Consider renaming etherRate to rateFactor, to accurately represent what it does.

PresaleLimit

The word limit, in the context, implies an upper bound, making it seem that the presale is restricted to sell only 10 ETH worth of tokens. In reality, this represents the minimum purchase value during the presale phase. Consider renaming it to presaleMinimum.

AMENDED [2018-1-14]:
This issue has been fixed by the Restart Energy team and is not present in `commit a89c78e`.

Closing Summary

Significant issues were found during the audit, some of which break the intended behaviour, while others are non-standard business decisions made. It's strongly advised that such issues are addressed before moving on with the ICO process.

RestartEnergyToken.sol has been verified as fully ERC20 compliant, and has taken recommended measures to mitigate the known [EIP20 API Approve / TransferFrom multiple withdrawal attack] (**https://github.com/ethereum/EIPs/issues/738**).

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 have been widely audited and secured, as such, they were not prioritized for auditing.

AMENDED [2018-1-14]:
All major issues reported in this audit have since been addressed by Restart Energy.

Contributors to the crowdsale should be aware that tokens can be distributed outside the standard purchase flow, by function `` `sendTokensToAddress` ``. Restart Energy has stated they will be using this functionality to distribute bonus tokens to private contributors. Since these distributions are logged through a `` `TokenPurchase` `` event with 0 logged as the contribution amount, they are easily identifiable/auditable. Consequently, the risk of this functionality being abused has been determined to be low.

Audit Report for Restart Energy ICO. January 13, 2018.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of the Restart Energy platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

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

Solidified Technologies Inc.