SOLIDIFIED

Audit Report for Iconiq Lab. March 07, 2018.

Summary

Audit Report prepared by Solidified for Iconiq Lab covering token and crowdsale contracts.

Process and Delivery

Two (2) independent Solidified experts performed an unbiased and isolated audit of the below contracts. The debrief took place on March 02, 2018 and the final results are presented here.

Audited Files

The following files were covered during the audit:

- * ICNQCrowdsale.sol (./contracts/ICNQCrowdsale.sol)
- * ICNQToken.sol (./contracts/ICNQToken.sol)
- * TeamAndAdvisorsAllocation.sol (./contracts/TeamAndAdvisorsAllocation.sol)
- * Whitelist.sol (./contracts/Whitelist.sol)

Intended Behavior

The purpose of these contracts is to create, distribute, and manage vesting for the tokens of the Iconiq Lab platform.

Issues Found

Critical Section

1. Incorrect tracking of token distribution across phases

ICNQCrowdsale uses `token.totalSupply()` (essentially the amount of tokens minted to date) throughout in an attempt to enforce the caps on token tranches; this intent is violated because the count (`totalSupply`) carries over from each phase to the next, but the checks do not take this into account.
This makes the following lines end their respective phases prematurely:

ICNQCrowdsale.sol / lines 127, 132

Potentially prevents pre- and main sales:

```
require(validPurchase() && token.totalSupply() <=
TOTAL_TOKENS_FOR_CROWDSALE);
```

Potentially ends pre-sale early:

```
require(token.totalSupply() <= PRE_SALE_TOTAL_TOKENS);
```

This will also behave incorrectly: the intent is to make sure no more than 3 million tokens have been sold in the main sale, but the check is done against the total amount of tokens minted again.

ICNQCrowdsale.sol / lines 147-148

```
//remainder logic
if (token.totalSupply().add(tokens) > TOTAL_TOKENS_FOR_CROWDSALE) {
```

This check will also not work as intended:
ICNQCrowdsale.sol / lines 170-172

```
if (token.totalSupply() == TOTAL_TOKENS_FOR_CROWDSALE) {
     return true;
}
```

Recommendation
Track progress of the tranches' sales separately, or set milestone constants to be absolute values instead of relative distances.

AMENDED [2018-3-7]:
This issue has been fixed by the Iconiq Lab team and is not present in commit `39bf9798338fc3a8bc23562ead0e2f38cac1fe75`.

2. Pre-sale and main sale purchases will fail

The largest effect of the above is `buyTokens` will always fail if more than 3 million tokens have been distributed in the private sale, since in function `buyTokens` checks that the total supply is not greater than `TOTAL_TOKENS_FOR_CROWDSALE` (3 million).

ICNQCrowdsale.sol / lines 27

```
require(validPurchase() && token.totalSupply() <=
TOTAL_TOKENS_FOR_CROWDSALE);
```

The private sale precedes the pre and main public sales, therefore the tokens minted to private buyers are included in `totalSupply` of the token contract. If more than 3 million tokens are distributed in this way (which is likely given that 10.15 million tokens are allocated for this purpose), the buy functionality will permanently fail.

AMENDED [2018-3-7]:
This issue has been fixed by the Iconiq Lab team and is not present in commit `39bf9798338fc3a8bc23562ead0e2f38cac1fe75`.

*** All following reports are made under the assumption that problem #1 has been fixed. ***

Major Section

3. The fact setTeamWalletAddress can be called at any time removes any team vesting guarantee

Because `setTeamWalletAddress` can be called even after the tokens have been sold out, owner can change his contract with the buyers after their payment. The best place for setting the address of TeamAndAdvisorsAllocation contract would probably be the constructor.

AMENDED [2018-3-7]:

After commit `39bf9798338fc3a8bc23562ead0e2f38cac1fe75` TeamAndAdvisorsAllocation can be set only once now, which is an improvement, but making sure it is set before public sale starts would be even better.

4. It's possible to buy at the bonus rate up to the total sale limit

If buy transaction gets processed before presale limit has been reached and results in surpassing the presale limit, buyer gets more tokens at the presale rate than intended, potentially up to the total presale token limit. This also breaks the refund logic, because it doesn't account for the possibility of buying at the presale rate.

Recommendation
It should be checked if the buy order exceeds the presale limit and either refund the excess ETH or reward the excess ETH at the standard rate.

AMENDED [2018-3-7]:
This issue has been fixed by the Iconiq Lab team and is not present in commit `39bf9798338fc3a8bc23562ead0e2f38cac1fe75`.

5. It's possible to call buyTokens function even after the crowdsale token limit has been reached

This condition:

```
require(validPurchase() && token.totalSupply() <=
TOTAL_TOKENS_FOR_CROWDSALE);
```

should read:

```
require(validPurchase() && token.totalSupply() <
TOTAL_TOKENS_FOR_CROWDSALE);
```

otherwise it's possible to call the buyTokens function even after the tokens have been sold out, overriding the `remainderAmount` and `remainderPurchaser` variable with each call.

AMENDED [2018-3-7]:

This issue has been fixed by the Iconiq Lab team and is not present in commit `39bf9798338fc3a8bc23562ead0e2f38cac1fe75`.

Minor Section

6. Refactor remainder logic

ICNQCrowdsale.sol / lines 148-155

Currently, the remainder refund only works for the main sale: pre-sale over-buys will just revert. Additionally, the crowdsale contract could transfer the `remainderAmount` immediately rather than rely on the contract authors; alternatively, the state variables could be replaced by logging an event to save gas, if the transfer remains manual.

```
if (token.totalSupply().add(tokens) > TOTAL_TOKENS_FOR_CROWDSALE) {
    tokens = TOTAL_TOKENS_FOR_CROWDSALE.sub(token.totalSupply());
    weiAmount = tokens.div(rate);

    // save info so as to refund purchaser after crowdsale's end
    remainderPurchaser = msg.sender;
    remainderAmount = msg.value.sub(weiAmount);
}
```

AMENDED [2018-3-7]:
This issue has been fixed by the Iconiq Lab team and is not present in commit `39bf9798338fc3a8bc23562ead0e2f38cac1fe75`.

Suggestion Section

7. Some mechanics of the crowdsale are not directly enforced in smart contract logic

Namely:
1. The gap between the end of the pre-sale and the start of the main sale, which is described in the intended behavior spec, will only occur if the contract owner pauses the sale. Otherwise, the sale will continue during this period.

2. That the unlock date of TeamAndAdvisorsAllocation is `April 1, 2019, 00:00:00`.
3. The refund of remainder in case of overbuy is expected to be performed manually.
4. Tokens from the `PRIVATE_SALE_TOTAL` pool that haven't been distributed through `mintTokenForPrivateInvestors` have to be manually burned, otherwise they become part of the public crowdsale pool.

Consider enforcing these in the smart contracts themselves.

8. Unexpected rate change can lead to different outcome than buyer expected when submitting transaction

If rate changes after buyer submits buy transaction, but before it gets included in blockchain, he can get rewarded with tokens at a different rate than expected.

Recommendation
This can be solved by adding argument to `buyTokens` function containing the desired rate, in case it doesn't match the current rate the function throws.

9. Other notes

- Variable `tokensIncludingBonus` on line ICNQToken.sol:142 is badly named, the name should be something like `bonusTokens`

- Burning unsold tokens by minting them to 0x0 unnecessarily inflates ICNQ's total supply, since finalization already stops further token minting. Consider removing the "burning" action on line ICNQToken.sol:188-193.

Closing Summary

Several critical issues were found during the audit, some of which could severely break the intended behaviour. Iconiq Team submitted the fixes for all Critical, Major and Minor issues found and performed a second audit, during which no additional issues were discovered. It is recommended to post the contracts on public bounty following the audit.

ICNQToken.sol has been verified as fully ERC20 compliant, and has taken recommended 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/etc. have been widely audited and secured and as such, were not prioritized for auditing.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of the Iconiq Lab 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.