SOLIDIFIED

Audit Report for Well, Inc. March 02, 2018.

Summary

Audit Report prepared by Solidified for Well, Inc. covering token and crowdsale 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 March 02, 2018 and the final results are presented here.

Audited Files

The following files were covered during the audit:

- Well.sol

Intended Behavior

The purpose of these contracts is to create the Well token and distribute it to the public, in a crowdsale process.

Issues Found

1. Owner can mint tokens as required

Owner can mint tokens during and after the crowdsale without any restrictions. As per the expected behaviour document, there should be a restriction on the token release dates for the remaining tokens. Minting the token during the crowdsale can also affect the total tokens available for purchase since the `totalSupply` should be less than `maxTokenToBuy` during the purchase.

Recommendation
It is recommended to add token release criterias for the team and bounty to the contract itself rather than handling it manually. Once the crowdsale is finished, the remaining tokens should be moved to a reserve rather than minting it as required. It is also recommended to restrict the owner from minting during the crowdsale.

2. WeiFounded is overwritten in every new purchase

Line 420 of Well.sol:

```
weiFounded = weiAmount;
```

This line, instead of adding the amount purchased to the total, is simply replacing it. This brings some problems like the hardcap is never reached.

Recommendation
Change line to

```
weiFounded = weiFounded.add(weiAmount);
```

AMENDED [2018-3-6]:
This issue has been fixed by the JoinWell team and is not present in the latest code shared.

3. Fiat hardcap can be exceeded by rate variance

With the current rate specified for each buy order, the tokens available for ICO can collect more ETH than the hard cap ($28M) specified in the provided document.

Recommendation

It is recommended to adjust the tokens available for sale or the rate in which tokens are sold to match the hardcap specified in the document.

4. Ether Hardcap can be bypassed

Line 379 of Well.sol

```
require(!icoFinished && weiFounded <= hardCap);
```

This line only checks if the current funded amount is lesser than the hardcap, but it does not consider the incoming transaction amount, which will surpass the hardcap.

Recommendation
Change this line to:

```
require(!icoFinished && weiFounded.add(msg.value) <= hardCap);
```

AMENDED [2018-3-6]:
This issue has been fixed by the JoinWell team and is not present in the latest code shared.

5. Token is susceptible to multiple withdraw attack

There is a known attack on ERC20 that allows an approved spender to transfer more than allowed by another user. Detailed description here.

Recommendation
Currently, the best practice is to leave the approve function as is, to preserve backwards compatibility and add another function for safely changing the approval amount.
Detailed discussion here.

AMENDED [2018-3-6]:
This issue has been fixed by the JoinWell team and is not present in the latest code shared.

6. Token calculation can overflow

A malicious owner can deny distributing tokens by increasing exponentially the 'rate' variable, such as when the code reaches line 413 it overflows, making the tokens variable very close to zero.

Recommendation
Make this calculation using the safeMath library. Another approach is to rethink owners ability to change rates arbitrarily.

7. ICO end date is not enforced by contract

The contract does not disallow purchases after the end date specified in the intended behaviour documentation, if the owner hasn't called the finishICO function.

Recommendation
Add a mechanism to reject purchases after the end date even if the owner does not trigger the finalizing function.

8. Bonus time period specified is different

Time period specified for bonus is different than the expected behaviour document provided. As per document, 40% bonus is provided between January 1 and March 15 (1521158400). In the smart contract, the specified period for 40% bonus is between January 1 and March 3 (1520121600).

```
if(date >= 1514764800 && date < 1520121600) { // Line 440 - 444
           bonusDate = 40;
} else if(date >= 1520121600 && date < 1523836800) {
            bonusDate = 30;
}
```

Recommendation

Change the time period for bonus to match the date provided in the document.

AMENDED [2018-3-6]:
This issue has been fixed by the JoinWell team and is not present in the latest code shared.

9. TokenPurchase event is not used

The defined event for tokens purchase is not used.

Recommendation
Trigger the event at the end of the buyTokens function

AMENDED [2018-3-6]:
This suggestion has been incorporated by the JoinWell team and is present in the latest code shared.

10. Consider removing enableTransfer

Owner can avoid enabling the token transaction after finishing the ICO. Consider enabling the transaction as soon as the ICO is finished or after a certain time period.

11. Older and non-fixed compiler version

The specified minimum compiler version is too old. Older compilers might be susceptible to some bugs. We recommend changing the solidity version pragma to the latest version (`0.4.20`) to enforce the use of an up to date compiler.

List of known compiler bugs and their severity can be found here.

AMENDED [2018-3-6]:
This suggestion has been partially incorporated by the JoinWell team. The current code shared enforces the use of minimum compiler version `0.4.20`.

12. Use of OpenZeppelin

OpenZeppelin contracts were copied from the repository. OpenZeppelin's MIT license requires the license and copyright notice to be included if its code is used. Use NPM to manage the library.

The contracts used are outdated versions of OpenZeppelin library. Update the libraries to the latest version which included input validations and other best practices.

AMENDED [2018-3-6]:
This suggestion has been partially incorporated by the JoinWell team. It is still recommended to use NPM to manage the files.

13. Consider changing the units used for tokens

Tokens are considered as Ethers in the contract. The representation is technically correct but may confuse the reader. Consider changing it to an actual token count to increase the readability.

14. Final transaction must be exact

The transaction will fail If the final transaction exceeds the remaining token amount. It is recommended to add an approach in which the remaining amount will be transferred back after purchasing the tokens.

15. Consider specifying visibility in the methods

It is recommended to add visibility specifiers such as public/private/internal/external to the function explicitly.

16. Consider using view/pure

Use view if your function does not modify storage and pure if it does not even read any state information.

Closing Summary

Several major issues were found during the audit, some of which can severely break the intended behaviour. It's strongly recommended that at the minimum all major and minor issues (color coded as orange and yellow) are corrected and then re-audited. It is furthermore recommended to post the contracts on public bounty afterwards.

SOLIDIFIED

Audit Report for Well, Inc. March 02, 2018.

Disclaimer

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