SOLIDIFIED

Audit Report for LandToken - June 14, 2021

Summary

Audit Report prepared by Solidified covering the LandToken smart contracts.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code in several rounds. The debrief took place on 14 May 2021.

Audited Files

The source code has been supplied in the form of a GitHub repository:

https://github.com/ls-jordan/LS---TEST-ROPSTEN

Commit number: `2ee391a13d94d0399fee9f192a68baffc7b393be`

The scope of the audit was limited to the following files:

```
contracts
├── LandToken.sol
├── LandTokenStake.sol
├── Migrations.sol
├── buyback.sol
└── stake.sol
```

Intended Behavior

The smart contracts implement a mintable and burnable ERC-20 token and a related staking solution.

Code Complexity and Test Coverage

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note, that high complexity or lower test coverage does equate to a higher risk. Certain bugs are more easily detected in unit testing than a security audit and vice versa. It is, therefore, more likely that undetected issues remain if the test coverage is low or non-existent.

Criteria	Status	Comment
Code complexity	Low	-
Code readability and clarity	High	-
Level of Documentation	Medium	Although no additional documentation has been provided the purpose of the code is clear from the inline commenting.
Test Coverage	N/A	No tests were submitted to the audit, so the test coverage could not be evaluated.

SOLIDIFIED

Audit Report for LandToken - June 14, 2021

Issues Found

Solidified found that the LandToken contracts contain no critical issues, 1 major issues, 4 minor issues, in addition to 4 informational notes.

We recommend all issues are amended, while the notes are up to the team's discretion, as they refer to best practices.

Issue #	Description	Severity	Status
1	BuyBack.sol and LandToken.sol: Contracts can lock ETH that cannot be extracted	Major	Pending
2	Order of arithmetic operation may decrease precision	Minor	Pending
3	LandToken.sol: Missing zero-check in mint role transfer	Minor	Pending
4	stake.sol: Ineffective access protection on read-only functions	Minor	Pending
5	stake.sol: Contract owner can manipulate stakers profits by temporarily changing currentAPR and landAPR values	Minor	Pending
6	Pragma allows for a wide range of compiler versions	Note	-
7	LandTokenStake.sol: no need to use SafeERC20 for LandToken	Note	-
8	stake.sol: Neither Withdraw nor Harvest events are emitted in the corresponding functions	Note	-
9	Contract.sol: Code Cleanup	Note	-

SOLIDIFIED

Audit Report for LandToken - June 14, 2021

Critical Issues

No critical issues have been found

Major Issues

1. BuyBack.sol and LandToken.sol: Contracts can lock ETH that cannot be extracted

The contract implements a payable `receive()` function. However, it does not implement any way to extract the funds, meaning that any value sent to the contract will be stuck forever. Similarly, `LandToken.sol` has a payable constructor, with no functionality for removing ETH from the contract.

Recommendation
Avoid receiving ETH or implement a way to recover it.

Minor Issues

2. Order of arithmetic operation may decrease precision

Throughout the codebase, multiplications are performed on the result of division. In integer arithmetic, this decreases the precision slightly. In some of these cases, the operations are performed over several lines of code and the small decrease in efficiency might be acceptable for improved readability. However, in other cases, the operation is performed in a single statement and reversal would not affect readability.

Recommendation
Consider reversing the order of operations to increase precision where appropriate.

3. LandToken.sol: Missing zero-check in mint role transfer

The contract allows for just one minter which can be transferred. However, there is no check for the `address(0)` in the `passMinterRole()` function. This means that minting functionality can be unintentionally renounced without option for recovery.

Recommendation
Add a zero check by adding:

```
require(contractMint != address(0));
```

4. stake.sol: Ineffective access protection on read-only functions

The view functions `getAmountWithdrawn()` and `getStakers()` perform the following check:

```
require(msg.sender==owner);
```

This check does not prevent reading these variables, since nothing on the blockchain is private. The values can be obtained easily bypassing these functions.

Recommendation
Remove ineffective checks.

5. stake.sol: Contract owner can manipulate stakers profits by temporarily changing currentAPR and landAPR values

Example 1: Contract owner can temporarily change `currentAPR` and `landAPR` to very high values, and `harvest()` huge APR, then change the `currentAPR` and `landAPR` back to normal values.
Example 2: Contract owner can temporarily change `currentAPR` and `landAPR` to 0, and call `storeHarves()` to prevent certain stakers from earning APR for the accumulated period. Then change the `currentAPR` and `landAPR` back to normal values.

Recommendation
Palace bounds on the values that can be set by the operator.

Informational Notes

6. Pragma allows for a wide range of compiler versions

The `pragma` statement allows for a very large range of compiler versions, including some versions with known bugs. In addition, the language syntax has changed since the earlier versions that are allowed.

Recommendation
Consider limiting the compiler to at least a single major version number.

7. LandTokenStake.sol: no need to use SafeERC20 for LandToken

The codebase uses `SafeERC20` to interact with `LandToken`. However, this is the platform's own trusted token and it's behavior is clear. The library can be removed in this instance in the interest of gas cost.

Recommendation
Consider removing `SafeERC20` to interact with `LandToken`.

8. stake.sol: Neither Withdraw nor Harvest events are emitted in the corresponding functions

The events `Withdraw` and `Harvest` are unused.

Recommendation
Consider emitting these events in the corresponding functions.

9. Contract.sol: Code Cleanup

It is strongly recommended to resolve the following items for a better code quality and readability.

1. Remove any unused variables. The contract contains many unused variables for example `LandTokenStake.S, LandTokensStake.SA, stake.thing` and `stake.actives`.
2. No need to associate a library with the type if it's not used. Remove the `SafeMath` association with `uint256` type if it's not used.
3. Remove all magic numbers, strings and addresses: The contract contains many hardcoded values which reduces the readability and increases the chance of error. It is recommended to use reusable variables for these.
4. The method `buyback.approveTokens` uses a very large number to simulate infinite approval. It is recommended to use `UINT_MAX` for such practices.
5. Duplicate code in `stake.sol` - It is recommended to extract the interest calculation to a method and reuse it where required.
6. Unwanted mapping in `stake.sol` - The mappings `harvestAmountBUSD` and `harvestAmountLand` is not required since the value is only used inside a method and is always set to 0. It is recommended to use local variables instead of state variables for them.

Audit Report for LandToken - June 14, 2021

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of LandToken or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. 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.