SOLIDIFIED

Audit Report for EZPOS. February 11, 2018.

Summary

Audit Report prepared by Solidified for EZPOS covering the EZToken contract.

Process and Delivery

Two (2) independent Solidified experts performed an unbiased and isolated audit of the below contract. The debrief took place on February 11, 2018 and the final results are presented here.

Audited Files

The following files were covered during the audit:

- [EZToken.sol]

Notes

The audit was performed on commit hash `2ccbd1432030353e679ba242eb8f4bfaed757da6` of github repo: https://github.com/ezpos/eztoken-contract.

Intended Behavior

The purpose of this contract is to create and distribute the EZToken.

Issues Found

1. Token is not ERC compliant

There are two issues that makes the EZToken non-compliant to the ERC20 standard (https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md).

- There's no implementation of `balanceOf` function
- There's no implementation of `allowance` function
- `Transfer` function does not returns true on success
- `Approval` event not implemented
- The constructor should trigger the Transfer event

The first and second items are present as variables, but that's not enough to satisfy the standard as the functions must be implemented.

Recommendation
Implement both functions, adjust the transfer interface and trigger the event on token creation.

AMENDED Feb 14, 2018:
issue fixed in commit 116d37fdf669bb4d3bebfcaf01eac27257948eba

2. Consider using re-entrancy guard

EZToken.sol | Line 178

Use re-entrancy guard for `approveAndCall()` to avoid any sort of attack from the spender contract. The method makes a call to an anonymous spender contract which may execute any malicious code. It is better to safeguard such methods from unexpected attacks.

Recommendation
Re-entrancy is avoided by setting a flag that allows the function to be called only once. The contract is available here:
https://github.com/OpenZeppelin/zeppelin-solidity/blob/master/contracts/ReentrancyGuard.sol.

AMENDED Feb 14, 2018:
issue fixed in commit 116d37fdf669bb4d3bebfcaf01eac27257948eba

3. Transaction-Ordering Dependence (TOD) in approve method

EZToken.sol | Line 163 & 178

Changing an allowance with `approve` and `approveAndCall` brings the risk that someone may use both the old and the new allowance by unfortunate transaction ordering.

Recommendation
One possible solution is to first reduce the spender's allowance to 0 and set the desired value afterwards.

More information is available here: https://github.com/ethereum/EIPs/issues/738
and here: https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729.

AMENDED Feb 14, 2018:
issue fixed in commit 116d37fdf669bb4d3bebfcaf01eac27257948eba

4. Chance of balance getting overwritten during the contract deployment

EZToken.sol | Line 50 - 99

If the same address is used in the constructor for different years, then the balance for that particular address will be overwritten during each assignment. For example, if the address `0x5a4...` is used for both `_year1` and `_year2`, then the final balance of the address will be 3.5M tokens and the balance will not be accessible during the freezing period of `_year2`.

There is a chance that this can be easily overlooked during the contract deployment. Consider solving this issue if it was not an intended behaviour. Also consider input validation with `require()`.

AMENDED Feb 14, 2018:
issue fixed in commit 116d37fdf669bb4d3bebfcaf01eac27257948eba

5. Consider using SafeMath

SafeMath (https://github.com/OpenZeppelin/zeppelin-solidity/blob/release/v1.5.0/contracts/math/SafeMath.sol) is a well-tested and popular library in solidity for performing arithmetic operations. It is suggested to use the library throughout the contract to protect from underflow/overflow.

AMENDED Feb 14, 2018:
issue fixed in commit 116d37fdf669bb4d3bebfcaf01eac27257948eba

6. Consider using OpenZepplin libraries for ERC20

It is suggested to use well-tested code to avoid potential security threats. OpenZepplin is a library with a set of reusable contracts that can be used here to achieve the ERC20 token functionalities.

The complete library is available [here](https://github.com/OpenZeppelin/zeppelin-solidity).

7. Gas optimization in the constructor

EZToken.sol | Line 50 - 99

Gas used can be further optimized in the constructor by avoiding redundant code. This helps in reducing the cost during the contract deployment.

For better readability, arrays can be used to manage the yearly token allocation.

AMENDED Feb 14, 2018:
issue fixed in commit 116d37fdf669bb4d3bebfcaf01eac27257948eba

8. Consider using the `'years'` alias

Solidity provides a global variable that accounts for years in unix time. Using this would make the time variables easier to read and much less error prone.

9. Older 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 (`^0.4.19`) as of this writing, to enforce the use of an up-to-date compiler.

List of known compiler bugs and their severity can be found here:
https://etherscan.io/solcbuginfo

AMENDED Feb 14, 2018:
The compiler version was not changed due to compatibility issues of version 0.4.19 with Mist.

10. Unused variable

EZToken.sol | Line 37

Unused variable `` `transfersEnabled` `` is declared in the contract. This can be removed.

If the plan is to enable and disable transfer, Pausable.sol (https://github.com/OpenZeppelin/zeppelin-solidity/blob/master/contracts/lifecycle/Pausable.sol) contract can be used to achieve the functionality.

AMENDED Feb 14, 2018:
issue fixed in commit 116d37fdf669bb4d3bebfcaf01eac27257948eba

11. Consider increasing the decimal case to 18.

The Ether has a 18 decimal cases and therefore much of tools developed for the ecosystem considers the same amount, and it's advised to keep your token in the same range unless you have a very strong reason to change it.

12. Consider Unifying `transfer` and `_transfer`

There's no reason to keep these two separate functions.

AMENDED Feb 14, 2018:
issue fixed in commit 116d37fdf669bb4d3bebfcaf01eac27257948eba

13. Use `constant`

Make use of `constant` if the value will not be changed later in the contract. It is a best practice to follow.

AMENDED Feb 14, 2018:
issue fixed in commit 116d37fdf669bb4d3bebfcaf01eac27257948eba

Closing Summary

3 Minor issues were found during the audit that could potentially break the desired behaviour. These issues along with most of the suggestions have been addressed by the EZPOS team and report was amended on Feb 14, 2018 to reflect the latest state.

Audit Report for EZPOS. February 11, 2018.

Disclaimer

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