SOLIDIFIED

Audit Report for SpringRole. January 23, 2018.

Summary

Audit Report prepared by Solidified for SpringRole covering the SRT Token contract.

Process and Delivery

One (1) independent Solidified expert performed an unbiased and isolated audit of the below token contract. The debrief took place on January 23, 2018 and the final results are presented here.

Audited Files

The following files were covered during the audit:

- Airdrop.sol
- WhitelistedContracts.sol
- SRTToken.sol

Intended Behavior

The set of contracts serves the purpose of creating and airdropping the SRT Token.

The provide intended behavior spec:

- Pre Mint Token
- Option to have white listed contracts withdraw tokens on behalf of user for transactions (doTransfer)
- Mint function to mint tokens up to 10B
- Cannot buy tokens with ETH
- Tokens are transferable

Issues Found

1. Max Supply can overflow

In the SRTToken.sol constructor the maxSupply variable can overflow, since SafeMath is not used there. If a sufficiently large number is passed as a parameter, the multiplication can give unreliable results.

Recommendation:

Change line 165 to:
```
maxSupply = _maxSupply.mul(10**decimals)
```

AMENDED [2018-1-24]:
This issue has been fixed by the SpringRole team and is not present in commit `6d0e68e`.

2. Token is susceptible to multiple withdrawal attack

That is a known attack on ERC20 that allows and approved spender to transfer more than allowed by another user. Detailed description here:
https://docs.google.com/document/d/1YLPtQxZu1UAvO9cZ1O2RPXBbT0mooh4DYKjA_jp-RLM

Recommendation:

Current, 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:
https://github.com/ethereum/EIPs/issues/738#issuecomment-336277632

AMENDED [2018-1-24]:
This issue has been fixed by the SpringRole team and is not present in commit `6d0e68e`.

3. Tokens are not assigned to owner in constructor

The comment in line 163 suggest that the owner should hold some balance after the execution is finished. However no alteration in the owner's balance is made in the function body.

Recommendation:

Assign some balance to owner in constructor or change comment to represent the actual function behaviour.

AMENDED [2018-1-24]:
This issue has been fixed by the SpringRole team and is not present in commit `6d0e68e`.

4. Total token supply can never reach maxSupply

The maximum reachable supply is maxSupply - 1 unit, since the mint function doesn't allow total supply to equals max Supply.

Recommendation:

Change line 193 to:

```
require (maxSupply >= (totalSupply.add(_amount)));
```

AMENDED [2018-1-24]:
This issue has been fixed by the SpringRole team and is not present in commit `6d0e68e`.

Suggestions

5. Consider decreasing the power given to whitelisted addresses

Whitelisted addresses have the power to remove tokens from anyone, regardless of authorization, and transfer to somebody else, by calling the doTransfer() function. This mechanism might reduce trust from token holders.

6. Consider replacing uint with bool in WhiteListedContracts

Instead of using 1 and 0 to indicate whether or not a contract is whitelisted, consider implementing a address to bool mapping.

Recommendation:

Change line 134 to:

```
mapping (address => bool ) white_listed_contracts;
```

AMENDED [2018-1-24]:
This issue has been fixed by the SpringRole team and is not present in commit `6d0e68e`.

7. Consider removing the payable fallback

If the contract is not expected to receive ether transactions, the best approach is to not have a payable function implemented, making all pure transactions fail and therefore, avoiding receiving unexpected ether.

Recommendation:
Remove the fallback function

AMENDED [2018-1-24]:
This issue has been fixed by the SpringRole team and is not present in commit `6d0e68e`.

Closing Summary

A few issues were found during the audit and some of them can break desired behaviour. It's strongly advised that these issues are corrected before moving on with the token process.

Disclaimer

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

Audit Report for SpringRole. January 23, 2018.