CRYPTOCURRENCY (HTTPS://BLOG.COINFABRIK.COM/CATEGORY/CRYPTOCURRENCY/)

Rightmesh Token Sale Smart Contract Audit (Master)

Ariel Yabo (https://blog.coinfabrik.com/author/ariel-yabo/)

March 6, 2018 (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/rightmesh-token-sale-smart-contract-audit-master/)

Coinfabrik was asked to audit the contracts for the RightMesh Token sale. In the first part, we will give a summary of our discoveries and follow them with the details of our findings.

The contracts audited are from the RightMesh repository at https://github.com/firstcoincom/solidity (https://github.com/firstcoincom/solidity). The audit is based on the commit ***f24ea6c5787b2d40423f4dc312d832592b1cd335*** at branch *master*.

Contents

Summary

The audited contracts are:

- MeshToken.sol: Token implementation.
- MeshCrowdsale.sol: Crowdsale implementation.
- Migrations.sol: Truffle migration contract.

These checks are performed:

- isuse of the several call methods: call.value(), send() and transfer().
- Integer rounding errors, overflow, underflow and related usage of SafeMath functions.
- Old compiler version pragmas.

- Race conditions such as reentrancy attacks or front running.
- Misuse of block timestamps, assuming things other than them being strictly increasing.
- Contract softlocking attacks (DoS).
- Potential gas cost of functions being over the gas limit.
- Missing function qualifiers or misuse of them.
- Fallback functions with higher gas cost than what a transfer or send call allows.
- Underhanded or erroneous code.
- Code and contract interaction complexity.
- Wrong or missing error handling.
- Overuse of transfers in a single transaction instead of using withdrawal patterns.
- Insufficient coverage of function input requirements.

Detailed findings

Critical/Medium severity

None found

Minor severity

Owner cannot be allowed to transfer on pause

There is an override of the **whenNotPaused** modifier which is used for all the token transfer related functions in OpenZeppelin implementation:

```
 modifier whenNotPaused() {

require(!paused || allowedTransfers[msg.sender]);

_;

 }
```

And **allowedTransfers** mapping can be modified by the owner, but not for himself:

```
 function updateAllowedTransfers(address _address, bool _allowedTransfers)

 external

 onlyOwner

 returns (bool)

 {

// don't allow owner to change this for themselves

// otherwise whenNotPaused will not work as expected for owner,

// therefore prohibiting them from calling pause/unpause.

require(_address != owner);



allowedTransfers[_address] = _allowedTransfers;

return true;

 }
```

We recommend removing this require, as the owner cannot pause the token twice:

(https://blog.coinfabrik.com/)

```
function pause() onlyOwner whenNotPaused public {

 require(pausedOnce == false);

 pausedOnce = true;

 super.pause();

}
```

(https://blog.coinfabrik.com/)

In order to stop depending on the behavior of **allowedTransfers**, you may also remove **whenNotPause** modifier from **pause** and move the requirement inside the function. Like this:

```
function pause() onlyOwner public {

 require(!paused && pausedOnce == false);

 pausedOnce = true;

 super.pause();

}
```

Update: *The RightMesh team declared they do not want the Owner to make token transactions.*

Function setLimit data race softlock

```
function setLimit(address[] _addresses, uint256 _weiLimit) external returns (bool) {

 require(whitelistingAgents[msg.sender] == true);



 for (uint i = 0; i < _addresses.length; i++) {

address _address = _addresses[i];



// only allow changing the limit to be greater than current contribution

require(_weiLimit >= weiContributions[_address]);

  weiLimits[_address] = _weiLimit;

 }

 return true;

}
```

This function does not set the **weiLimit** for an address if said limit is bigger than its contributions. This can be exploited by buying tokens just before this function is called, to get over the limit which will cause the function to fail. Without even considering a hypothetical attack, this function may get softlocked indefinitely by multiple people contributing.

This can be fixed by directly setting the maximum value between the current **weiContribution** for the address and the new **weiLimit**. For example, assuming max is already implemented:

```
function setLimit(address[] _addresses, uint256 _weiLimit) external returns (bool) {

 require(whitelistingAgents[msg.sender] == true);

 for (uint i = 0; i < _addresses.length; i++) {

address _address = _addresses[i];

  weiLimits[_address] = max(_weiLimit, weiContributions[_address]);

 }

 return true;

}
```

Update: *The RightMesh team declared they will never use this function to reduce weiLimits, only to increase them which avoids the issue.*

Outdated and differing solidity pragma versions

The three files are using different old Solidity versions. **Migrations.sol** and **MeshToken.sol** are using **0.4.17**, while **MeshCrowdsale.sol** is using **0.4.15**. Consider updating them all to the same latest version of Solidity.

Update: *This was fixed in commit 59c1977301c071e3cd7fac670b00aa51d87bf1aa*

Enhancements

Remove unnecessary pause and unpause functions from Crowdsale

The following functions in the crowdsale are not needed:

```
/**

* @dev Allows the contract owner to pause the token transfers on deployed token

*/

function pauseToken() external onlyOwner {

 MeshToken(token).pause();

}

/**

* @dev Allows the contract owner to unpause the token transfers on deployed token

*/

function unpauseToken() external onlyOwner {

 MeshToken(token).unpause();

}
```

These functions are made redundant by **transferTokenOwnership**:

```
/**                                                    (https://blog.coinfabrik.com/)
* @dev Allows the current owner to transfer token control back to contract owner
*/            (https://blog.coinfabrik.com/)
function transferTokenOwnership() external onlyOwner {
 token.transferOwnership(owner);
}
```

Which allows the owner to call the **pause** and **unpause** functions in the token directly.

Update: *Fixed in commit c5454aa92583b21273764cc22ae595209b46c92e*

Conclusion

The contracts were simple, they both relied on OpenZeppelin's implementation of the token and sale. Using an already implemented, tested and audited token is always the best option if feasible. Neither contract added overcomplicated or large elements to the implementations which is also a desirable approach.

Do you want to know what is Coinfabrik Auditing Process?

Check our A-Z Smart Contract Audit Guide (https://blog.coinfabrik.com/smart-contract-audits-ultimate-security-guide/) or you could request a quote (https://www.coinfabrik.com#contact_us) for your project.

Related Posts

(https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/dmtoken-token-sale-smart-contract-audit/)

DMToken Token Sale Smart Contract Audit (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/dmtoken-token-sale-smart-contract-audit/)

Coinfabrik was hired to audit the contract in terms of its security. First of all,...

(https://blog.coinfabrik.com/)

(https://blog.coinfabrik.com/)

(https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/realisto-token-sale-smart-contract-audit/)

Realisto Token Sale Smart Contract Audit (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/realisto-token-sale-smart-contract-audit/)

By Ismael Bejarano and Pablo Yabo Coinfabrik smart contract audit's team was hired to review...

(https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/dmtoken-token-sale-smart-contract-audit/)

DMToken Token Sale Smart Contract Audit (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/dmtoken-token-sale-smart-contract-audit/)

Coinfabrik was hired to audit the contract in terms of its security. First of all,...

Tags:

audit (https://blog.coinfabrik.com/tag/audit/)

ERC20 (https://blog.coinfabrik.com/tag/erc20/)

Rightmesh (https://blog.coinfabrik.com/tag/rightmesh/)

Token (https://blog.coinfabrik.com/tag/token/)

SHARE ON

(https://www.facebook.com/sharer/sharer.php?u=https://blog.coinfabrik.com/smart-contracts/smart-co
smart-contracts/rightmesh-token-sale-smart-contract-audit-master/)

(https://twitter.com/intent/tweet?
text=Rightmesh%20Token%20Sale%20Smart%20Contract%20Audit%20(Master)&url=https://blog.coinfab
contracts/smart-contract-audit-smart-contracts/rightmesh-token-sale-smart-contract-audit-master/)

(https://pinterest.com/pin/create/button/?url=&media=https://blog.coinfabrik.com/wp-
content/uploads/2018/03/rightmesh.png&description=Rightmesh+Token+Sale+Smart+Contract+Audit+%

(https://www.linkedin.com/shareArticle?mini=true&url=https://blog.coinfabrik.com/smart-contracts/smar
smart-contracts/rightmesh-token-sale-smart-contract-audit-
master/&title=Rightmesh%20Token%20Sale%20Smart%20Contract%20Audit%20(Master)&source=CoinFa

← PREVIOUS ARTICLE

NEXT ARTICLE →
(https://blog.coinfabrik.com/)

Beluga Pay (BBI) Security Audit (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/beluga-pay-bbi-security-audit/)
(https://blog.coinfabrik.com/)

Security Audit – Send (SDT) Token Sale ICO Smart Contract (https://blog.coinfabrik.com/smart-contracts/security-audit-send-sdt-token-sale-ico-smart-contract/)

You may also like

(https://blog.coinfabrik.com/smart-contracts/magic-bridge-audit/)

Magic Bridge Audit (https://blog.coinfabrik.com/smart-contracts/magic-bridge-audit/)

(https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/mintingfactoryv2-baseupgradablemarketplace-kodav3upgradablegatedmarketplace/)

2 months ago

Smart Contract Audit (https://blog.coinfabrik.com/category/smart-contracts/smart-contract-audit-smart-contracts/)

MintingFactoryV2, BaseUpgradableMarketplace & KODAV3UpgradableGatedMarketplace (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/mintingfactoryv2-baseupgradablemarketplace-kodav3upgradablegatedmarketplace/)

(https://blog.coinfabrik.com/feed/)

(https://ar.linkedin.com/company/coinfabrik)

(https://twitter.com/coinfabrik)

(https://www.youtube.com/channel/UC2GmjCr7aEzil31kqOy9aw) (https://blog.coinfabrik.com/)

(https://www.facebook.com/CoinFabrik/) (https://blog.coinfabrik.com/)

(https://www.reddit.com/r/CoinFabrik/)

(https://github.com/coinfabrik)