Audit for RexToken ICO Contract

Author: Matthew Di Ferrante

Summary

Generally, the contract looks good. There's only one major issue I would consider a "security risk", which is the migrate function, as it has no time limits.

Beyond that, constant variables are well formed. Only recommendation would be to construct TOTAL_SHARE as follows:

TOTAL_SHARE = CROWDSALE_SHARE + ANGELS_SHARE + CORE_1_SHARE + CORE_2_SHARE + PARTNERSHIP_SHARE + REWARDS_SHARE + AFFILIATE_SHARE

Instead of assigning it 1000, in case you want to change any of the share distributions right before the crowdsale, prevents mental math mistakes.

Main minor security recommendation is the use of 'transfer' instead of 'send', and initializing totalSupply explicitly inside of RexToken.

Please read through the "description", "recommendation" and "security risk / issue" sections for each function below

version() constant returns (bytes32)

description

Returns version string. Constant result, no arguments.

issues

none

RexToken(uint256 _start, address _vault)

description

Constructor, sets startTime and vault from arguments, sets isFinalized to false.

issues

none

recommendation:

- set isFinalized to false state when declaring the variable instead of in the constructor
- initialize totalSupply here instead of relying on inheritance values

[default] () payable

description

default function, just a passthrough to createTokens

issues

none

checkInvariant() returns(bool)

description

bounty function from openzeppelin code - empty

issues

none

createTokens(address recipient) payable

uses state from

```
- msg.value
- startTime
- DURATION
- weiRaised
- WEI_RAISED_CAP
- msg.sender
- getRate()
- totalSupply
- balances
- vault
```

modifies

```
+ weiRaised
+ totalSupply
+ balances
+ vault
```

description

```
main function that issues tokens,
checks that the wei sent is greater than 0,
checks that the time is between startTime and startTime + DURATION
ensures weiRaised is less than WEI_RAISED_CAP
if wei sent + wei raised is greater than WEI_RAISED_CAP, return the excess wei
if the send fails, throw
tokens issued = bonus rate from getRate multiplied by accepted wei amount
add tokens to totalSupply
add wei accepted to weiRaised
update balance for recipient with safemath
shoot off TokenCreated event
send wei to vault, throw if fail
```

recommendation

Use transfer instead of send in this function. Also, Currently there's no reentrancy vulnerability, as the amount of gas needed to get to the first send is at least 2099 so a reentrant call would OOG before completing. Still, for best practices, weiRaised should be added to before the first msg.sender.send call.

getRate() constant returns (uint256)

uses state from

```
- startTime
- BASE_RATE
```

description

```
for the first week, rate is BASE_RATE + 300
for the second week, rate is BASE_RATE + 200
for the third week, rate is BASE_RATE + 100
after that, rate is BASE_RATE
```

issues

none

tokenAmount(uint256 share, uint256 finalSupply) constant returns (uint)

uses state from

```
- TOTAL_SHARE
```

description

```
returns 'share' argument multiplied by 'finalSupply', divided by TOTAL_SHARE
or (finalSupply/TOTAL_SHARE) * share
```

issues

none

grantTokensByShare(address to, uint256 share, uint256 finalSupply) internal

uses state from

```
- tokenAmount()
- balances
- totalSupply
```

modifies

```
+ balances
+ totalSupply
```

description

```
internal function called only by finalize,
adds token balances to shareholders from _SHARE constants in the contract,
fires off TokenCreated event
adds assigned tokens to totalSupply
```

issues

none

migrate(uint256 amount)

uses state from:

```
- msg.sender
- 0x00fdbabeb74187ef64bea1bf56821f61fd93ab3990
- balances
- totalSupply
```

security risk:

This function has no time limits, it should not be callable after the token sale is over. At least, there should be a way to disable it in case the old contract is found to have a bug, or in case the old contract's behaviour is mutable.

getFinalSupply() constant returns (uint256)

uses state from:

```
- TOTAL_SHARE
- totalSupply
- CROWDSALE_SHARE
```

description

```
returns finalSupply by taking totalSupply * (TOTAL_SHARE/CROWDSALE_SHARE)
(currently returns totalSupply * 2)
```

issues

none

finalize() onlyOwner

uses state from

```
- isFinalized
- weiRaised
- WEI_RAISED_CAP
- startTime
- DURATION
- getFinalSupply()
```

modifies:

```
+ isFinalized
```

modifies through

```
+ grantTokensByShare()
```

description

```
finalizes the crowdsale,
if isFinalized is set to false
and |weiRaised over the cap OR sale is over|
get final crowdsale-raised supply

using finalSupply calculatin, grant tokens shares to:
    ANGELS      -  5%
    CORE_1      -  7.5%
    CORE_2      -  7.5%
    PARTNERSHIP -  7%
    REWARDS     -  20%
    AFFILIATE   -  3%

(totaling 50% - other 50% is crowdsale share)

set isFinalized to true
```

issues

none