Security Audit of BCND

Conclusion

Audit was done by the “Web3Go” team https://web3go.tech/ (https://web3go.tech/)
by Vladimir Smelov vladimirfol@gmail.com (mailto:vladimirfol@gmail.com)
https://www.linkedin.com/in/vladimir-smelov-25021669/ (https://www.linkedin.com/in/vladimir-smelov-25021669/)

In the final contract were not found:

- Backdoors for investor funds withdrawal by anyone.
- Bugs allowing to steal money from the contract.
- Other security problems.

Obvious errors or backdoors were not found in the contract.
The client was acknowledged about all secutiry notes below.

Scope

https://github.com/bitcluster-ru/bcnd-smart-contract (https://github.com/bitcluster-ru/bcnd-smart-contract)
commit: `d5ffaabb`

```
$ md5sum contracts.*
6ecb49f196873001aefce8a385cd723d  contracts/BitClusterNordCrowdsale.sol
7571465dc5f8e6e6b5fde56c1a3458f5  contracts/BitClusterNordToken.sol
04fb5bde9db46ab4661121d2ec90c91f  contracts/ERC20PresetOwnablePausable.sol
```

Methodology

1. Blind audit. Try to understand the structure of the code.
2. Find info in internet.
3. Ask quiestions to developers.
4. Draw the scheme of cross-contracts interactions.
5. Write user-stories, usage cases.
6. Run static analyzers

Find problems with:

- backdoors
- bugs
- math
- potential leaking of funds
- potential locking of the contract
- validate arguments and events
- others

Result

Critical

Major

1. Infinity minting.

At:

- contracts/ERC20PresetOwnablePausable.sol:29
  Owner is able to mint as many coins as he wants at any time.
  This is not reliable.

Recommendation.

Add comment why it should be possible.
Or add the flag `_mintingFinished` and method `setMintingFinished` to disable minting forever after a while.

Client's comment.

> This was done deliberately. Specfics of the project are such that additional tokens may need to be minted indefinitely, and inhibiting minting in the future may be actively harmful.

Status.

ACKNOWLEDGED

2. Return value is ignored

Call ignores return value by token.transfer

- contracts/BitClusterNordCrowdsale.sol#122
  ERC20 requires to check transfer success status.

Recommendation.

Use SafeERC20 library.
Or check success status manually.

Status.

FIXED at d5f0b6237d30cbe6083afd069f5330276c57c17c

Warning

1. Address zero-check for attribute set.

At

- contracts/BitClusterNordCrowdsale.sol#52-54
- contracts/BitClusterNordCrowdsale.sol#132

it’s not checked that the address is not 0.
It may be broken in front-end.
It’s a good practice to check address to be not address(0).

Recommendation.

Add

```
require(_address != address(0), "ZERO_ADDRESS");
```

Client’s comment.

> I agree that these checks should be added, but unfortunately adding them breaks our test coverage report for some reason. I will debug it later, but we are on a tight schedule and I doubt that we will be able to fix this before deployment.
> These methods are not called from the frontend anyway, they are admin-only, so the risk is acceptable. We will perform manual testing of contract functionality after deployment to ensure that no zero addresses creeped in.

Status.

ACKNOWLEDGED

2. Multiplication on the result of a division

At

- contracts/BitClusterNordCrowdsale.sol#82

```
msg.value * uint256(ethUsdExchangeRate) /
(10 ** ethUsdExchangeRateFeed.decimals()) * rate
```

It decreases the accuracy of the calculation.

Recommendation.

First perform multiplications, then all divisions.

```
msg.value * uint256(ethUsdExchangeRate) * rate /
(10 ** ethUsdExchangeRateFeed.decimals())
```

Status.

FIXED at d5f0b6237d30cbe6083afd069f5330276c57c17c

Comment

1. Methods should be declared external.

Everywhere where method is used only externaly (never internaly) it’s better to set modifer to `external` not `public` to save gas.

```
- BitClusterNordCrowdsale.remainingSupply() (contracts/BitClusterNordCrowdsale.
- BitClusterNordCrowdsale.pause() (contracts/BitClusterNordCrowdsale.sol#140-14
- BitClusterNordCrowdsale.unpause() (contracts/BitClusterNordCrowdsale.sol#148-
- ERC20PresetOwnablePausable.mint(address,uint256) (contracts/ERC20PresetOwnabl
- ERC20PresetOwnablePausable.pause() (contracts/ERC20PresetOwnablePausable.sol#
- ERC20PresetOwnablePausable.unpause() (contracts/ERC20PresetOwnablePausable.so
```

Recommendation.

Make methods `external` to save gas.

Status.

FIXED at d5f0b6237d30cbe6083afd069f5330276c57c17c

2. Storing BTC address in dynamicly sized string.

At:

- contracts/BitClusterNordToken.sol:13
  you use string to store btc address, however the BTC address size is 26-35 alphanumeric values, which you can effectively pack into bytes32 (even smaller but it's not required since solidity will use one bytes32 memory storage slot anyway)

See also:

- https://medium.com/layerx/how-to-reduce-gas-cost-in-solidity-f2e5321e0395#1b1b (https://medium.com/layerx/how-to-reduce-gas-cost-in-solidity-f2e5321e0395#1b1b)
- https://mudit.blog/solidity-gas-optimization-tips/ (https://mudit.blog/solidity-gas-optimization-tips/)

Recommendation.

Use bytes32 to save gas.
You can include methods encodeBTCAddressToBytes decodeBTCAddressFromBytes for simplicity.

Client's comment.

> I'm against this. This hugely increases the internal and external complexity of the contract, with possible additional bugs, and the gas cost of deploying the encoder-decoder functions will likely be higher than expected gas savings from reduced storage.

Status.

ACKNOWLEDGED
NOT_ISSUE

3. Redundant variable.

At:

- contracts/BitClusterNordToken.sol:37
  It makes no sense to copy storage variable into the local copy.

Recommendation.

Rewrite as:

```
        IERC20(tokenAddress).safeTransfer(to, amount);
```

Status.

FIXED at d5f0b6237d30cbe6083afd069f5330276c57c17c

4. Redundant statement.

At:

- contracts/BitClusterNordCrowdsale.sol:15
  The statement is redundant because IERC20Metadata inherits from IERC20.

Recommendation.

Remove the statement.

Status.

FIXED at d5f0b6237d30cbe6083afd069f5330276c57c17c

5. Redundant usage of usdt.decimals.

At:

- contracts/BitClusterNordCrowdsale.sol:101
  Since you know that usdt decimals is a constant you can skip the
  `10**(18-usdt.decimals())`
  at all.

Recommendation.

Remove the statement.

Client's comment

> This was done deliberately. We are using YEENUS token as a substitute for USDT on test networks, and it has different decimals (8). It's important for us to be able to deploy the exact same versions of smart-contract code both in production and in testing (removing extra pre-deployment tweaks), so we deemed the gas cost of extra .decimals() call acceptable.

Status.

ACKNOWLEDGED
NOT_ISSUE

6. Do not use transfer to send ethers.

Read:

- https://solidity-by-example.org/sending-ether/ (https://solidity-by-example.org/sending-ether/)

Recommendation.

Use modern recommended way to send ethers.

Status.

FIXED at d5f0b6237d30cbe6083afd069f5330276c57c17c

7. Declare variable immutable.

At:

- contracts/BitClusterNordCrowdsale.sol:22
- contracts/BitClusterNordCrowdsale.sol:25
- contracts/BitClusterNordCrowdsale.sol:19
- contracts/BitClusterNordCrowdsale.sol:37
- contracts/BitClusterNordCrowdsale.sol:39
  It's better to use `immutable` declaration for such variables to save gas and to ensure the variable to be immutable.

Proof of concept:

```
contract A {
    uint256 immutable public value;
    uint256 public other;
    constructor (uint256 _value) {
        value = _value;
    }
    function stub(uint256 x) external {
        other = x*value;
    }
}

contract B {
    uint256 public value;
    uint256 public other;
    constructor (uint256 _value) {
        value = _value;
    }
    function stub(uint256 x) external {
        other = x*value;
    }
}
```

check gas:

```
def test_gas(admin):
    a = A.deploy(2, {'from': admin})
    b = B.deploy(2, {'from': admin})
    tx_a = a.stub(2, {'from': admin})
    tx_b = b.stub(2, {'from': admin})
    tx_a.info()
    tx_b.info()
```

output with gas used:

```
Function: A.stub
Block: 31
Gas Used: 41522 / 40000000 (0.1%)


Function: B.stub
Block: 32
Gas Used: 42325 / 40000000 (0.1%)
```

Recommendation.

use

```
uint256 immutable public endTime;
```

Status.

FIXED at d5f0b6237d30cbe6083afd069f5330276c57c17c