BLOCKCHAIN FOUNDRY

Baanx.com

Smart Contract Audit Report

June 18th, 2021

BLOCKCHAIN FOUNDRY

info@blockchainfoundry.co
www.blockchainfoundry.co

Baanx Token Generation Project:

Project Synopsis

Baanx is building the future of modular, API driven, Financial Services that include debit cards, digital wallets, IBANs, remittance/FX, payment gateways and more.

The Baanx team contracted Blockchain Foundry to perform a smart contract audit on their preexisting ERC20 smart contract.

The contract we reviewed was the rinkeby contract:
https://rinkeby.etherscan.io/address/0x6992A1c8E8657e83C6f17870B282071883C52637

We were also made aware of the live contract during the process::
https://etherscan.io/token/0x54f9b4b4485543a815c51c412a9e20436a06491d#readContract

We note the findings apply to both rinkeby as well as mainnet contract deployments.

Findings

All findings were non-critical issues that should not affect the performance of the logic in the contract however may affect standardization and costs incurred by users as read below.

1. Evm Optimizations Turned off - this will incur higher gas costs and generally can be turned on.

2. No need to directly import and use SafeMath in SOLC 8 - Solidity 8.0 handles math internally using SafeMath. You can save deployment fees here but it looks like it has already been deployed. Just for future reference.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/utils/math/SafeMath.sol";

contract BXXToken is ERC20 {
    using SafeMath for uint256;

    uint256 constant private INITIAL_AMOUNT_WHOLE_TOKENS = 250e6;

    constructor (string memory name, string memory symbol) ERC20(name, symbol) {
        _mint(
            msg.sender,
            INITIAL_AMOUNT_WHOLE_TOKENS * (10 ** uint256(decimals()))
        );
    }

    function burn(address account, uint256 amount) external {
        _burn(account, amount);
    }
}
```

3. Use ERC20Burnable instead of implementing a custom burn wrapper function - The Open Zeppelin ERC20Burnable interface exposes the burn function and should be used to inherit from instead of ERC20 directly.

Conclusion

Generally due to the assumption that the underlying interfaces are safe under the General Theory of System Safety then this contract is completely safe for its intended uses as a general ERC20 contract. You may use this report to upload to the Etherscan contract audit feature as required.