Big Doge Coin Security Audit

Big Doge Coin security audit, conducted by the Callisto Network Security Department during July 2021.

Big Doge Coin Security Audit Report

Are Your Funds Safe?

Audit Request

Our main focus is to provide the best entertainment possible for our players on Blockchain. We place key emphasis on creating games that are of a high quality in every way possible. This means that we ensure that all our games themes are engaging, our features unique and captivating and our art, stunning and precise! We don't cut corners and do whatever is necessary to offer a memorable game-play experience.

- Website: ~~https://bigdoge.io/~~
- Twitter: https://twitter.com/bigdogecrypto
- Community: https://t.me/bigdogecoin
- Channel: https://t.me/bigdogenews
- Instagram: https://www.instagram.com/bigdogecoin/
- Facebook: https://www.facebook.com/Big-Doge-Coin-102179458805235
- Youtube: https://www.youtube.com/channel/UCP6HVzQf3ClG9hgZEPFz7tA

- Reddit: https://www.reddit.com/user/bigdogecrypto

Source code

https://bscscan.com/address/0x2BA8c3066F36B998bC74CE8DcE260Fb5D2ba6bCc#code

Disclosure policy

Standard disclosure policy.

Platform

BSC.

1. In scope

- https://etherscan.io/address/0x0aff88b4cf3015c9c17f1da1fccb88c632f3505e#code

2. Findings

In total, **1 issue** were reported including:

- 0 high severity issue.
- 0 medium severity issue.
- 1 low severity issue.

In total, **9 notes** were reported, including:

- 3 notes.
- 6 owner privilege.

2.1 Possible incorrect message in `require` function.

Severity: note.

Description:

In the line 466 `require(now > _lockTime , "Contract is locked until 7 days");` the message indicates a specific locking period, but function `function lock()` (line 456) allow to lock to any period.

Recommendation:

Replace with message without specific locking period.

2.2 No excluded accounts – unused code.

Severity: note.

Description:

The variables in lines 701-702 is declared but never initialized and there are no functions to initialize them.

```
mapping (address => bool) private _isExcluded;
address[] private _excluded;
```

Therefore contract could not have excluded from rewards addresses, and numbers of functions and its part are unused:

1. Part of code in function `balanceOf` (line 791)
2. Function `isExcludedFromReward` (lines 825-827)
3. Part of code in function `deliver` (line 835)
4. Part of code in function `_takeLiquidity` (lines 951-952)
5. Part of code in function `_tokenTransfer` (lines 1101-1109)
6. Function `_transferToExcluded` (lines 1126-1144)
7. Function `_transferFromExcluded` (lines 1136-1144)
8. Function `_transferBothExcluded` (lines 860-869)
9. Part of code in function `_getCurrentSupply` (lines 938-943).

Recommendation:

To increase readability and reduce deployment cost the unused code should be removed.

2.3 ERC20 Complines – transfer 0 value should be allowed

Severity: low.

Description:

Due ERC20 standard: `Transfers of 0 values MUST be treated as normal transfers and fire the Transfer event.`

Recommendation:

Remove `require(amount > 0, "Transfer amount must be greater than zero");` (line 1001).

2.4 Unused variable `mintedByDxsale`

Severity: note.

Description:

The variable `mintedByDxsale` (line 708) was declared but never used.

Recommendation:

To increase readability and reduce deployment cost the unused code should be removed.

2.5 Owner privileges

Severity: owner privileges.

Description:

Contract owner has right to:

1. Exclude/include any account from/in fee, using functions `excludeFromFee` (lines 871-873) and `includeInFee` (lines 875-877).
2. Change tax fee in range from 0 to `maxTaxFee` percent, using function `setTaxFeePercent` (lines 879-878).
3. Change liquidity fee in range from 0 to `maxLiqFee` percent, using function `setLiquidityFeePercent` (lines 884-887).
4. Change maximal amount per transaction in range from `minMxTxPercentage` to 100 percent of total supply, using function `setMaxTxPercent` (lines 889-894).
5. Enable or disable adding liquidity to pool, using function `setSwapAndLiquifyEnabled` (lines 896-899).
6. Enable or disable all fees and transaction amount limit, using functions `disableFees` (lines 1146-1154) and `enableFees` (lines 1156-1161).

3. Security practice

- ☑ **Open-source contact**.
- ☐ **The contract should pass a bug bounty after the completion of the security audit.**
- ☐ **Public testing.**
- ☐ **Automated anomaly detection systems.** – NOT IMPLEMENTED. A simple anomaly detection algorithm is recommended to be implemented to detect behavior that is atypical compared to normal for this contract. For instance, the contract must halt deposits in case a large amount is being withdrawn in a short period of time until the owner or the community of the contract approves further operations.
- ☐ **Multisig owner account.**
- ☐ **Standard ERC20-related issues.** – NOT IMPLEMENTED. It is known that every contract can potentially receive an unintended ERC20-token deposit without the ability to reject it even if

the contract is not intended to receive or hold tokens. As a result, it is recommended to implement a function that will allow extracting any arbitrary number of tokens from the contract.

- ☐ **Crosschain address collisions.** ETH, ETC, CLO, etc. It is possible that a transaction can be sent to the address of your contract at another chain (as a result of a user mistake or some software fault). It is recommended that you deploy a “mock contract” that would allow you to withdraw any tokens from that address or prevent any funds deposits. Note that you can reject transactions of native token deposited, but you can not reject the deposits of ERC20 tokens. You can use this source code as a mock contract: extractor contract source code. The address of a new contract deployed using `CREATE (0xf0)` opcode is assigned following this scheme `keccak256(rlp([sender, nonce]))`. Therefore you need to use the same address that was originally used at the main chain to deploy the mock contract at a transaction with the `nonce` that matches that on the original chain. *Example: If you have deployed your main contract with address 0x010101 at your 2021th transaction then you need to increase your nonce of 0x010101 address to 2020 at the chain where your mock contract will be deployed. Then you can deploy your mock contract with your 2021th transaction, and it will receive the same address as your mainnet contract.*

4. Conclusion

The audited smart contract can be deployed. Only low severity issues were found during the audit.

It is recommended to adhere to the security practices described in pt. 4 of this report to ensure the contract’s operability and prevent any issues that are not directly related to the code of this smart contract.

Appendix

Smart Contract Audits by Callisto Network.

Miscellaneous

Why Audit Smart Contracts?

Our Most Popular Audit Reports.

Trust the Blockchain, Audit the Smart Contracts.

Follow Callisto’s Security Department on Twitter to get our latest news and updates!

Published on **July 20, 2021**

Security Audits

‹ Previous post

Next post ›

Callisto Network LTD

71-75 Shelton Street
London, Greater London
United Kingdom, WC2H 9JQ

Join Our Community

Resources

FAQ
Timeline
Airdrop
Community Guidelines

Callisto

Partners
Our GitHub repositories
Media Kit
Contact us
Want to sell your CLO coins OTC?