Security Assessment

DogeX

Oct 4th, 2021

Table of Contents

Summary

This report has been prepared for DogeX to discover issues and vulnerabilities in the source code of the DogeX project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	DogeX
Platform	Ethereum
Language	Solidity
Codebase	
Commit	

Audit Summary

Delivery Date	Oct 04, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

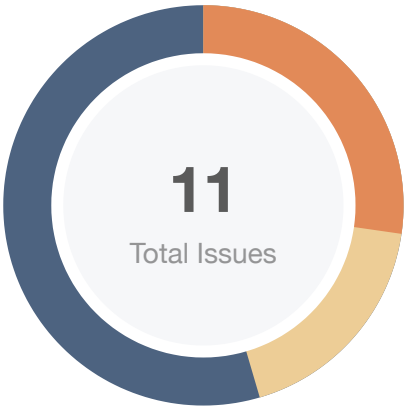
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	3	0	0	3	0	0
Medium	0	0	0	0	0	0
Minor	2	0	0	2	0	0
Informational	6	0	0	6	0	0
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
DXD	DogeX-main/DogeX.sol	bb2a7940d931c0603e1c85249186a7ce8421d763a775e2d11d931e0330946e11

Findings



Critical	0 (0.00%)
Major	3 (27.27%)
Medium	0 (0.00%)
Minor	2 (18.18%)
Informational	6 (54.55%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
DXD-01	Potential Sandwich Attacks	Logical Issue	Minor	Acknowledged
DXD-02	Proper usage of public and private type	Coding Style	Informational	Acknowledged
DXD-03	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
DXD-04	Fee Distribution	Logical Issue	Informational	Acknowledged
DXD-05	Owner can add an address as a bot arbitrarily	**Centralization / Privilege**	**Major**	Acknowledged
DXD-06	3rd party dependencies	Control Flow	Minor	Acknowledged
DXD-07	Centralized risk in `addLiquidity`	**Centralization / Privilege**	**Major**	Acknowledged
DXD-08	Variable could be declared as `constant`	Gas Optimization	Informational	Acknowledged
DXD-09	Return value not handled	Volatile Code	Informational	Acknowledged
DXD-10	Missing event emitting	Coding Style	Informational	Acknowledged
DXD-11	Missing check for the valid input	Logical Issue	Informational	Acknowledged

DXD-01 | Potential Sandwich Attacks

Category	Severity	Location	Status
Logical Issue	Minor	DogeX-main/DogeX.sol: 641~653	Acknowledged

Description

A sandwich attack might happen when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (before the transaction being attacked) a transaction to purchase one of the assets and make profits by backrunning (after the transaction being attacked) a transaction to sell the asset.

The following function is called without setting restrictions on slippage or minimum output amount, so transactions triggering by the function are vulnerable to sandwich attacks, especially when the input amount is large:

- `getTokenPriceBNB()`

Recommendation

We recommend setting reasonable minimum output amounts, instead of 0, based on token prices when calling the aforementioned functions.

DXD-02 | Proper usage of public and private type

Category	Severity	Location	Status
Coding Style	Informational	DogeX-main/DogeX.sol: 429~435	Acknowledged

Description

The following private state variables are only visible for the contract they are defined in. Even though it might be potential design, we recommend the team to let users be aware of the following addresses.

- `_marketingAddress`
- `_buybackAddress`
- `_projectMaintainence`
- `_developmentAddress`
- `presaleRouter`
- `presaleAddress`

Recommendation

We recommend to use `public` attribute.

DXD-03 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	DogeX-main/DogeX.sol	Acknowledged

Description

In the contract `DogeX`, the role `owner` has the authority over the following function:

- `excludeFromFee()`
- `includeInFee()`
- `removeBot()`
- `addBot()`
- `excludeFromContractWallet()`
- `includeInContractWallet()`
- `includeInExchange()`
- `excludeFromExchange()`
- `includeInBridge()`
- `excludeFromBridge()`
- `setMaxTxAmount()`
- `setMaxWalletAmount()`
- `setPercents()`
- `setTaxes()`
- `setPriceImpact()`
- `setManualETHvalue()`
- `updateOraclePriceFeed()`
- `setPresaleRouterAndAddress()`
- `endPresale()`
- `enablePriceOracle()`
- `disablePriceOracle()`
- `setFloor()`
- `setFloorPercent()`
- `updateTaxFreeBlocks()`
- `updatePairSwapped()`
- `updateMinBuySellBNB()`
- `updateMaxSellAmountBNB()`
- `enableSellCoolDown()`

- `disableSellCoolDown()`
- `enableDailyMax()`
- `disableDailyMax()`
- `setFloorFees()`
- `setSlideFees()`
- `updateBuyBackAddress()`
- `updateMarketingAddress()`
- `updateDevelopmentAddress()`

In the contract `DogeX`, the role `_marketingAddress` has the authority over the following function:

- `manualswap()`
- `manualsend()`

Any compromise to the `owner` and `_marketingAddress` account may allow the hacker to take advantage of these.

Recommendation

We advise the client to carefully manage the `owner` account's and `_marketingAddress` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

DXD-04 | Fee Distribution

Category	Severity	Location	Status
Logical Issue	Informational	DogeX-main/DogeX.sol: 343~377, 393~396, 574~583	Acknowledged

Description

There're six types of taxes depending on the situation: `BuybackFee`, `MarketingFee`, `LiquidityFee`, `ReflectionFee`, `DevFee` and `_projectMaintainencePercent`. All above tax rates are different under four different scenes: buying, selling, golden hour 1 and golden hour 2. The details of fees are listed as below. In the buy scenario:

- _buyBuybackFee = %5;
- _buyMarketingFee = %3;
- _buyReflectionFee = %2;
- _buyLiquidityFee = %3;
- _buyDevFee = %1;

In the sale scenario:

- _sellBuybackFee = %7;
- _sellMarketingFee = %3;
- _sellReflectionFee = %2;
- _sellLiquidityFee = %4;
- _sellDevFee = %1;

In golden hour 1:

- _floorBuybackFee = %21;
- _floorMarketingFee = %7;
- _floorLiquidityFee = %5;
- _floorReflectionFee = %4;
- _floorDevFee = %3;

In golden hour 2:

- _slideBuybackFee = %13;
- _slideMarketingFee = %5;
- _slideLiquidityFee = %3;
- _slideReflectionFee = %2;

- _slideDevFee = %1;

If the transaction comes from a bot address or is sent to a bot address, in the buy scenario,

- _buyMarketingFee = 45;
- _buyBuybackFee = 45;
- _buyReflectionFee = 0;
- _buyDevFee = 0;

in the sale scenario,

- _sellMarketingFee = 45;
- _sellBuybackFee = 45;
- _sellReflectionFee = 0;
- _sellDevFee = 0;

60% of fees will be sent to the `_buybackAddress`, 25% will be used in marketing, 10% will be sent to the `_developmentAddress`, and the rest 5% will be used for project maintenance.

Recommendation

This is the business logic of the DogeX protocol, however, users should be aware of the fee distribution.

DXD-05 | Owner can add an address as a bot arbitrarily

Category	Severity	Location	Status
Centralization / Privilege	**Major**	DogeX-main/DogeX.sol: 1115~1117	Acknowledged

Description

Owner can add an address as a bot arbitrarily.

Recommendation

We recommend the team listing the conditions for making an address to a bot.

DXD-06 | 3rd party dependencies

Category	Severity	Location	Status
Control Flow	Minor	DogeX-main/DogeX.sol	Acknowledged

Description

The contract is serving as the underlying entity to interact with third party PancakeSwap protocols. The scope of the audit would treat those 3rd party entities as black boxes and assume its functional correctness. However in the real world, 3rd parties may be compromised that led to assets lost or stolen.

Recommendation

We understand that the business logic of the DogeX protocol requires the interaction PancakeSwap protocol for adding liquidity to DogeX-BNB pool and swap tokens. We encourage the team to constantly monitor the statuses of those 3rd parties to mitigate the side effects when unexpected activities are observed.

DXD-07 | Centralized risk in `addLiquidity`

Category	Severity	Location	Status
Centralization / Privilege	Major	DogeX-main/DogeX.sol: 859~871	Acknowledged

Description

```
// add the liquidity
uniswapV2Router.addLiquidityETH{value: ethAmount}(
    address(this),
    tokenAmount,
    0, // slippage is unavoidable
    0, // slippage is unavoidable
    owner(),
    block.timestamp
);
```

The `addLiquidity` function calls the `uniswapV2Router.addLiquidityETH` function with the `to` address specified as `owner()` for acquiring the generated LP tokens from the `DogeX-BNB` pool. As a result, over time the `_owner` address will accumulate a significant portion of LP tokens.If the `_owner` is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Recommendation

We advise the `to` address of the `uniswapV2Router.addLiquidityETH` function call to be replaced by the contract itself, i.e. `address(this)`, and to restrict the management of the LP tokens within the scope of the contract's business logic. This will also protect the LP tokens from being stolen if the `_owner` account is compromised. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;
- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

DXD-08 | Variable could be declared as `constant`

Category	Severity	Location	Status
Gas Optimization	Informational	DogeX-main/DogeX.sol	Acknowledged

Description

Variables `_threshold` could be declared as `constant` since this state variable is never to be changed.

Recommendation

We recommend declaring the variable as `constant`.

DXD-09 | Return value not handled

Category	Severity	Location	Status
Volatile Code	Informational	DogeX-main/DogeX.sol: 859~871	Acknowledged

Description

The return values of function `addLiquidityETH` are not properly handled.

```
        uniswapV2Router.addLiquidityETH{value: ethAmount}(
            address(this),
            tokenAmount,
            0, // slippage is unavoidable
            0, // slippage is unavoidable
            owner(),
            block.timestamp
        );
```

Recommendation

We recommend using variables to receive the return value of the functions mentioned above and handle both success and failure cases if needed by the business logic.

DXD-10 | Missing event emitting

Category	Severity	Location	Status
Coding Style	Informational	DogeX-main/DogeX.sol	Acknowledged

Description

In contract `DogeX`, there are a bunch of functions can change state variables. However, these function do not emit event to pass the changes out of chain.

Recommendation

Recommend emitting events, for all the essential state variables that are possible to be changed during runtime.

DXD-11 | Missing check for the valid input

Category	Severity	Location	Status
Logical Issue	Informational	DogeX-main/DogeX.sol: 1103~1109	Acknowledged

Description

The above functions don't check the account to see if it's already included/excluded from fee.

Recommendation

We recommend performing checks before excluding/including the account from fee.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.