AI Doge

Smart Contract Audit Report

SYSFIXED

22 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
AI Doge	DogeGPT	BSC

Addresses

Contract address	0xd1b8894E05F336f8dA2F7a594DC48e890038d341
Contract deployer address	0x4bE827b3eF1B238247D65Bf5212692c174cB40e9

Project Website

https://aidogebsc.com/

Codebase

https://bscscan.com/address/0xd1b8894E05F336f8dA2F7a594DC48e890038d341#code

SUMMARY

AiDoge is a meme project with a long foundation, with a large professional team, and the support of large projects like Rocket Raccoon and Binance. We'll be the hottest meme of 2023. The first Doge on Binance Smart Chain using Artificial Intelligence for the people and by the people.

Contract Summary

Documentation Quality

This project has a standard of documentation.

- Technical description provided.

Code Quality

The quality of the code in this project is up to standard.

- The official Solidity style guide is followed.

Test Scope

Project test coverage is 100% (Via Codebase).

Audit Findings Summary

Issues Found

- SWC-101 | Arithmetic operation issues discovered on lines 21, 31, 40, 41, 51, 408, 427, and 504.
- SWC-103 | A floating pragma is set on line 7, the current pragma Solidity directive is ""^0.8.15"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.
- SWC-108 | State variable visibility is not set on line 405. It is best practice to set the visibility of state variables explicitly to public or private.
- SWC-110 | Out of bounds array access issues discovered on lines 589 and 590.

CONCLUSION

We have audited the AI Doge project which has released on January 2023 to discover issues and identify potential security vulnerabilities in AI Doge Project. This process is used to find technical issues and security loopholes that find some common issues in the code.

The security audit report produced satisfactory results with low-risk issues.

The most common issue found in writing code on contracts that do not pose a big risk is that writing on contracts is close to the standard of writing contracts in general. The low-level issue found is a floating pragma is set, state variable visibility is not set, and out of bounds array access which the index access expression can cause an exception in case of use of an invalid array index value.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	ISSUE FOUND
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Check-Effect Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	PASS
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Caller	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS

Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	PASS
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique Id	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	PASS
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS

SMART CONTRACT ANALYSIS

Started	Sat Jan 21 2023 21:25:13 GMT+0000 (Coordinated Universal Time)
Finished	Sun Jan 22 2023 00:01:33 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	DogeGPT.Sol

Detected Issues

ID	Title	Severity	Status
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 21

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- DogeGPT.Sol

Locations

```
20   function add(uint a, uint b) internal pure returns (uint) {
21   uint c = a + b;
22   require(c >= a, "SafeMath: addition overflow");
23
24   return c;
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 31

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- DogeGPT.Sol

Locations

```
30   require(b <= a, errorMessage);
31   uint c = a - b;
32
33   return c;
34   }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 40

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- DogeGPT.Sol

Locations

```
39
40   uint c = a * b;
41   require(c / a == b, "SafeMath: multiplication overflow");
42
43   return c;
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 41

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- DogeGPT.Sol

Locations

```
40   uint c = a * b;
41   require(c / a == b, "SafeMath: multiplication overflow");
42
43   return c;
44   }
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 51

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- DogeGPT.Sol

Locations

```
50   require(b > 0, errorMessage);
51   uint c = a / b;
52
53   return c;
54   }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 408

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- DogeGPT.Sol

Locations

```
407
408   uint256 public numTokensSellToFee = 100000 * 10**18;
409
410   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
411   event SwapAndLiquifyEnabledUpdated(bool enabled);
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 427

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- DogeGPT.Sol

Locations

```
426   _owner = msg.sender ;
427   _totalSupply = 1000000000 * (10**18);
428
429   _balances[_owner] = _totalSupply;
430
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 504

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- DogeGPT.Sol

Locations

```
503   {
504   require(_numTokensSellToFee >= 10000 * 10**18 && _numTokensSellToFee <= 10000000 *
10**18, "Threshold must be set within 10,000 to 10,000,000 tokens");
505   numTokensSellToFee = _numTokensSellToFee;
506   }
507
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 7

low SEVERITY

The current pragma Solidity directive is ""^0.8.15"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- DogeGPT.Sol

Locations

```
6
7   pragma solidity ^0.8.15;
8
9   interface IBEP20 {
10   function totalSupply() external view returns (uint);
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 405

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "inSwapAndLiquify" is internal. Other possible visibility settings are public and private.

Source File

- DogeGPT.Sol

Locations

```
404
405   bool inSwapAndLiquify;
406   bool private swapAndLiquifyEnabled = true;
407
408   uint256 public numTokensSellToFee = 100000 * 10**18;
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 589

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- DogeGPT.Sol

Locations

```
588   address[] memory path = new address[](2);
589   path[0] = address(this);
590   path[1] = uniswapV2Router.WETH();
591
592   _approve(address(this), address(uniswapV2Router), tokenAmount);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 590

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- DogeGPT.Sol

Locations

```
589   path[0] = address(this);
590   path[1] = uniswapV2Router.WETH();
591
592   _approve(address(this), address(uniswapV2Router), tokenAmount);
593
```

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 Sysfixed'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 Sysfixed 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 is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

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.

This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Sysfixed and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Sysfixed) owe no duty of care.

ABOUT US

Sysfixed is a blockchain security certification organization established in 2021 with the objective to provide smart contract security services and verify their correctness in blockchain-based protocols. Sysfixed automatically scans for security vulnerabilities in Ethereum and other EVM-based blockchain smart contracts. Sysfixed a comprehensive range of analysis techniques—including static analysis, dynamic analysis, and symbolic execution—can accurately detect security vulnerabilities to provide an in-depth analysis report. With a vibrant ecosystem of world-class integration partners that amplify developer productivity, Sysfixed can be utilized in all phases of your project's lifecycle. Our team of security experts is dedicated to the research and improvement of our tools and techniques used to fortify your code.