STAYSAFU
AUDIT

MAY 1st, 2022

SPACE DOGE COIN

TABLE OF CONTENTS

I. SUMMARY

II. OVERVIEW

III. FINDINGS

- A. UINT-1 : Wrong uint size
- B. MSG-2 : Missing error message
- C. MSG-3 : Too long error message
- D. COMP-1 : Unfixed version of compiler
- E. BLOC-1 : Use of block.timestamp
- F. TX-1 : Use of tx.origin
- G. CENT-1 : Centralization of major privileges
- H. EXT-1 : External protocol dependance

IV. GLOBAL SECURITY WARNINGS

V. DISCLAIMER

AUDIT SUMMARY

This report was written for Space Doge Coin (SPACEDOGE) in order to find flaws and vulnerabilities in the Space Doge Coin project's source code, as well as any contract dependencies that weren't part of an officially recognized library.

A comprehensive examination has been performed, utilizing Static Analysis, Manual Review, and Space Doge Coin Deployment 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
- Through line-by-line manual review of the entire codebase by industry expert

AUDIT OVERVIEW

PROJECT SUMMARY

Project name	SPACE DOGE COIN
Description	Space Doge Coin is a non-existent USDT reflection distribution token. As long as you hold Space Doge Coin tokens, a 15-35% tax (100%) will be converted into USDT and distributed proportionally to all Space Doge Coin holders.
Platform	Binance Smart Chain
Language	Solidity
Codebase	https://bscscan.com/address/0xf54822a56640b2d44196aa6f42fc6200f2175f30#code

FINDINGS SUMMARY

Vulnerability	Total
Critical	0
Major	0
Medium	2
Minor	3

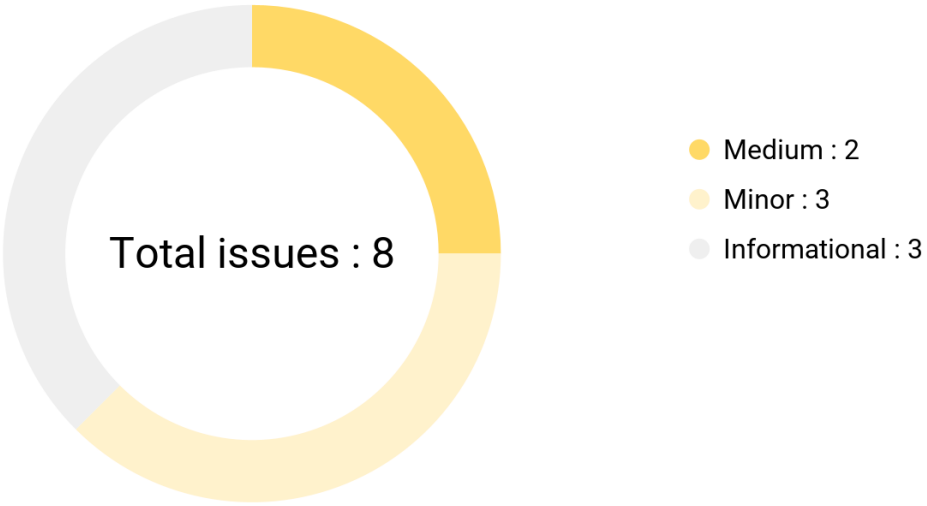
Informational	3

EXECUTIVE SUMMARY

Space Doge Coin is a non-existent USDT reflection distribution token. Purchases are taxed at 10% (100% of this tax is redistributed to the token holders). Sales are taxed between 10% and 30%, depending on the period during which the token is sold. The earlier you sell the token, the higher the sale fees will be. 50% of the tokens will be paid to Elon Musk if he manifests himself by sending a message on twitter. The token includes a self reflection and dividends mechanism to reward holders.

There have been no major or critical issues related to the codebase and all findings listed here are minor or informational. The major security problems are the dependence on a decentralized exchange platform and the centralization of privileges.

AUDIT FINDINGS



Code	Title	Severity
CENT-1	Centralization of major privileges	Medium
EXT-1	External protocol dependencies	Medium
BLOC-1	Usage of block.timestamp	Minor
TX-1	Usage of tx.origin	Minor
COMP-1	Unfixed version of compiler	Minor
UINT-1	Wrong uint size	Informational
MSG-2	Missing error message	Informational
MSG-3	Too long	Informational

UINT-1 | Wrong uint size

Description :

Some variables in the contract are of type uint, but not of the right size.

In order to optimize gas costs when deploying and using the contract,

We invite you to assign the right size uint to each variable.

8 errors of this type have been found in the smart contract.

Recommendation :

We recommend changing these uint sizes :

```
//Edited code containing appropriate uint sizes.
//l1348
uint8[] public rewardsFee = [0,0,0,0]; //(since the sum
of all these fees is under 100, every single fee is under
100 so you can use uint8)
//l 1351
uint32 public gasForProcessing = 300000; //(since
gasForProcesssing is under 500000)
//l 1375
event GasForProcessingUpdated(
    uint32 indexed newValue,
    uint32 indexed oldValue
);
//1535
function updateClaimWait(uint32 claimWait) external
onlyOwner {
    ...
}
```

```
//l 1539
function getClaimWait() external view returns (uint32) {
    ...
}
//l 1688
uint8 fees; //(a fee variable cannot be that high that it
needs being over 256)
//l 1774
uint32 public claimWait;
//l 1816
function updateClaimWait(uint32 newClaimWait) external
onlyOwner {
    ...
}
```

MSG-2 | Missing error message

Description :

Some require statements of the smart contract guarantee validity of conditions but do not throw any error message if these conditions are not met. We recommend documenting errors as best as possible in order to improve their handling and the user interface.

2 issues of this type have been found in the smart contract.

Recommendation :

We recommend adding an error message to the following require statement :

```
//Edited code containing missing error messages
//l 1207
require(totalSupply() > 0, "Cannot distribute if supply
is 0");
//l 1807
require(!isExcludedFromDividends[account], "Account
excluded from dividends");
```

MSG-3 | Too long error messages

Description :

The smart contract has some error messages that are too long. The industry standards specify error messages must have a maximal length of 32 bytes. We recommend having the shortest possible error messages to optimize gas costs (see github.com/ethereum/solidity/issues/4588) and improve error handling. 20 issues of this type have been found in the smart contract.

Recommendation :

We recommend shortening these error messages :

```
//Edited code containing shortened error messages
//l 378
require(currentAllowance >= amount, "Transfer amount over
allowance");
//l 419
require(currentAllowance >= subtractedValue, "Cannot
decrease allowance < 0");
//l 446
require(sender != address(0), "Transfer from the 0
address");
//l 447
require(recipient != address(0), "Transfer to the 0
address");
//l 452
require(senderBalance >= amount, "Transfer amount over
```

```
balance");
//l 501
require(accountBalance >= amount, "Burn amount exceeds
balance");
//l 530
require(owner != address(0), "Approve from the 0
address");
//l 531
require(spender != address(0), "Approve to the 0
address");
//l 1489
require(_sell.length == rewardsFee.length-1, "Wrong
amount of sell fees");
//l 1500
require(
    pair != uniswapV2Pair,
    "The pancakeswap pair cannot be removed"
);
//l 1509
require(
    automatedMarketMakerPairs[pair] != value,
    "Market maker pair is set to that value"
    );
//l 1523
require(
    newValue >= 200000 && newValue <= 500000,
    "newValue < 200000 or newValue > 500000"
    );
//l 1527
require(
    newValue != gasForProcessing,
    "gasForProcessing already set to this value"
    );
```

```
//l 1649
require(from != address(0), "Transfer from the 0
address");
//l 1650
require(to != address(0), "Transfer to the 0 address");
//l 1651
require(balanceOf(from) >= amount, "Transfer amount over
balance");
//l 1796
require(false, "Transfers unallowed");
//l 1800
require(
false,
"Dividend_Tracker: withdrawDividend disabled. Use the
'claim' function on the main ERC20DividendToken
contract."
); //This one is very long but we think that shortening
it would take away too much information
//l 1817
require(
    newClaimWait >= 1 && newClaimWait <= 86400,
    "claimWait < 1 second or > 24 hours"
);
//l 1821
require(
     newClaimWait != claimWait,
    "claimwait already set at this value"
    );
```

COMP-1 | Unfixed version of compiler

Description :

SPACEDOGECOIN token's contract does not have locked compiler versions, meaning a range of compiler versions can be used. This can lead to differing bytecodes being produced depending on the compiler version, which can create confusion when debugging as bugs may be specific to a specific compiler version(s).

11 errors of this type have been found in the smart contract.

Recommendation :

We fully understand that the Space Doge Coin code requires several contracts written with different versions of solidity. However, we encourage you to standardize the compiler versions as much as possible to avoid this problem.

BLOC-1 | Using block.timestamp

Description :

The use of block.timestamp can be problematic. The timestamp can be partially manipulated by the miner (see https://cryptomarketpool.com/block-timestamp-manipulation-attack/). In this smart contract this is not critical as in the worst case an attacker could only manipulate timings for liquidity exchanges and auto-claiming mechanisms.

Recommendation :

We fully understand the smart contract's logic of the SPACEDOGECOIN token. The use of block.timestamp is required to power the auto-liquify mechanism and we cannot replace it. Nevertheless, it is still useful to point out this kind of potential security problem.

TX-1 | Use of tx.origin

Description :

The use of tx.origin is strongly deprecated by the industry. It can lead to phishing attacks by falsifying the identity of the original caller of the function. For more information, see https://blockchain-academy.hs-mittweida.de/courses/solidity-coding-beginners-to-intermediate/lessons/solidity-5-calling-other-contracts-visibility-state-access/topic/tx-origin-and-msg-sender/ . Here it's not a security problem since tx.origin is used to send events. However, we still want to point out that the use of tx.origin is not recommended.

CENT-1 | Centralization of major privileges

Description :

The onlyOwner modifier of the smart contract gives major privileges over it (owner can set the automated market pair and set balance for a specific account)*. This can be a problem, in the case of a hack, an attacker who has taken possession of these privileged accounts could damage the project and the investors.

*This list is not exhaustive but presents the most sensitive points

Recommendation :

We recommend at least to use a multi-sig wallet as owner address, and at best to establish a community governance protocol to avoid such centralization. For more information, see https://solidity-by-example.org/app/multi-sig-wallet/

EXT-1 | Dependence to an external protocol

Description :

The contract is serving as the underlying entity to interact with third party PancakeSwap protocols. The scope of the audit would treat this third party entity as black box and assume it is fully functional. However in the real world, third parties may be compromised and may have led to assets lost or stolen.

Recommendation :

We encourage the team to constantly monitor the security level of the entire PancakeSwap project, as the security of the token is highly dependent on the security of the decentralized exchange platform.

Global security warnings

These are safety issues for the whole project. They are not necessarily critical problems but they are inherent in the structure of the project itself. Potential attack vectors for these security problems should be monitored.

CENT-1 | Global SPOF (Single Point Of Failure)

The project's smart contracts often have a problem of centralized privileges. The onlyOwner system in particular can be subject to attack. To address this security issue we recommend using a multi-sig wallet, establishing secure project administration protocols and strengthening the security of project administrators.

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 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 without StaySAFU's prior written consent.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 StaySAFU 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.

StaySAFU's position is that each company and individual are responsible for their own due diligence and continuous security. StaySAFU'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 fun.