SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

Doge Coin

AUDIT

SECURITY ASSESSMENT

05. August, 2024

FOR

Introduction

SolidProof.io is a brand of the officially registered company MAKE Network GmbH, based in Germany. We're mainly focused on Blockchain Security such as Smart Contract Audits and KYC verification for project teams.
Solidproof.io assess potential security issues in the smart contracts implementations, review for potential inconsistencies between the code base and the whitepaper/documentation, and provide suggestions for improvement.

Disclaimer

SolidProof.io reports are not, nor should be considered, an "endorsement"or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc'...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing 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. SolidProof's position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of the security or functionality of the technology we agree to analyze.

Project Overview

Summary

Project Name	Doge coin
Website	https://dogecoin.app/
About the project	$DOGE will be launched on Ethereum, BNB Chain, Arbitrum, and Base. Buy and Stake now during the Presale to maximize your rewards before the price skyrockets!
Chain	Ethereum
Language	Solidity
Codebase	https://etherscan.io/address/0x06106ec4b9cd940258862855a14c3960b2b8d276#code
Commit	N/A
Unit Tests	Not Provided

Social Medias

Telegram	https://t.me/DogeCoinL2
Twitter	https://x.com/DogeCoinL2
Facebook	N/A
Instagram	N/A
GitHub	N/A
Reddit	N/A
Medium	N/A
Discord	N/A
YouTube	N/A
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Change Log
v1.0	05. August 2024	· Layout Project · Automated/ Manual-Security Testing · Summary

Note – The following audit report presents a comprehensive security analysis of the smart contract utilized in the project that includes outside manipulation of the contract's functions in a malicious way. This analysis did not include functional testing (or unit testing) of the contract/s logic. We cannot guarantee 100% logical correctness of the contract as we did not functionally test it. This includes internal calculations in the formulae used in the contract.

File Overview

The Team provided us with the files that should be tested in the security assessment. This audit covered the following files listed below with an SHA-1 Hash.

File Name	SHA-1 Hash
contracts/PresaleV2.sol	baf462f80a979b2d10ffd5e3e85be5246ef67e4f

Please note: Files with a different hash value than in this table have been modified after the security check, either intentionally or unintentionally. A different hash value may (but need not) be an indication of a changed state or potential vulnerability that was not the subject of this scan.

Imported packages.

Used code from other Frameworks/Smart Contracts.

Dependency / Import Path	Count
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol	1

Note for Investors: We only audited contracts mentioned in the scope above. All contracts related to the project apart from that are not a part of the audit, and we cannot comment on its security and are not responsible for it in any way.

External/Public functions

External/public functions are functions that can be called from outside of a contract, i.e., they can be accessed by other contracts or external accounts on the blockchain. These functions are specified using the function declaration's external or public visibility modifier.

State variables

State variables are variables that are stored on the blockchain as part of the contract'sstate. They are declared at the contract level and can be accessed and modified by any function within the contract. State variables can be needed within visibility modifier, such as public, private or internal, which determines the access level of the variable.

Components

📝 Contracts	📚 Libraries	🔍 Interfaces	🎨 Abstract
1	0	2	0

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐 Public	💰 Payable
32	2

External	Internal	Private	Pure	View
28	31	0	0	6

StateVariables

Total	🌐 Public
24	23

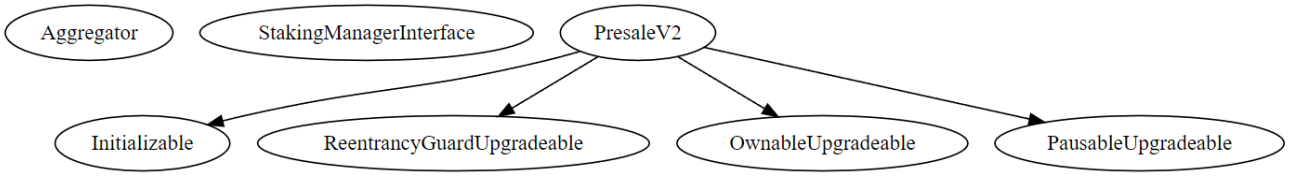
Capabilities

Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
^0.8.9	----------	Yes		----------

Transfers ETH	Low-Level Calls	Delegate Call	Uses Hash Functions	ECRecover	New/Create/Create2
yes					

Inheritance Graph

An inheritance graph is a graphical representation of the inheritance hierarchy among contracts. In object-oriented programming, inheritance is a mechanism that allows one class (or contract, in the case of Solidity) to inherit properties and methodsfrom another class. It shows the relationships between different contracts and how they are related to each other through inheritance.



Audit Information

Vulnerability & Risk Level

Risk represents the probability that a certain source threat will exploit the vulnerability and the impact of that event on the organization or system.The risk level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon aspossible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executingthe contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to check the repository for security-related issues, code quality, and compliance with specifications and best practices. To this end, our team of experienced pen-testers andsmart contract developers reviewed the code line by line and documented any issues discovered.

We check every file manually. We use automated tools only so that they help us achieve faster and better results.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
   a. Reviewing the specifications, sources, and instructions provided to
   SolidProof to ensure we understand the size, scope, and functionality of the
   smart contract.
   b. Manual review of the code, i.e., reading the source code line by line to identify potential vulnerabilities.
   c. Comparison to the specification, i.e., verifying that the code does what is described in the specifications, sources, and instructions provided to SolidProof.

2. Testing and automated analysis that includes the following:
   a. Test coverage analysis determines whether test cases cover code and how much code is executed when those test cases are executed.
   b. Symbolic execution, which is analysing a program to determine what inputs cause each part of a program to execute.

3. Review best practices, i.e., review smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on best practices, recommendations, and research from industry and academia.

4. Concrete, itemized and actionable recommendations to help you secure your smart contracts.

Overall Security

Upgradeability

Contract is not upgradable	✗ Deployer can update the contract with new functionalities.
Description	The contract is an upgradeable contract. The Deployer is not able to change or add any functionalities to the contract after deploying.
Comment	The contract contains the functionality in which the deployer of the contract can update and deploy the new version of the contract.

File/line(s): L71-92

Codebase: PresaleV2.sol

```
trace | funcSig
function initialize(
    address _oracle,
    address _usdt,
    uint256 _startTime,
    uint256 _startPrice,
    address _paymentWallet
) external initializer {
    require(_oracle != address(0), "Zero aggregator address");
    require(_usdt != address(0), "Zero USDT address");
    require(_startTime > block.timestamp, "Invalid time");
    __Pausable_init_unchained();
    __Ownable_init_unchained();
    __ReentrancyGuard_init_unchained();
    startTime = _startTime;
    startPrice = _startPrice;
    baseDecimals = (10 ** 18);
    aggregatorInterface = Aggregator(_oracle);
    USDTInterface = IERC20Upgradeable(_usdt);
    paymentWallet = _paymentWallet;
    timeConstant = 1 days;
    emit SaleTimeSet(startTime, block.timestamp);
}
```

Ownership

Contract ownership is not renounced.	❌ **The ownership is not renounced.**
Description	The owner has not renounced the ownership that means that the owner retains control over the contract's operations, including the ability to execute functions that may impact the contract's users or stakeholders. This can lead to several potential issues, including: • Centralizations • The owner has significant control over contract's operations.
Comment	N/A

Note – *The contract cannot be considered as renounced till it is not deployed or having some functionality that can change the state of the contract.*

Ownership Privileges

These functions can be dangerous. Please note that abuse can lead to financial loss. We have a guide where you can learn more about these Functions.

Minting tokens

Minting tokens refer to the process of creating new tokens in a cryptocurrency or blockchain network. This process is typically performed by the project's owner or designated authority, who has the ability to add new tokens to the network's total supply.

Contract owner cannot mint new tokens.	✅ **The owner cannot mint new tokens.**
Description	The owner is not able to mint new tokens once the contract is deployed.
Comment	N/A

Burning tokens

Burning tokens is the process of permanently destroying a certain number of tokens, reducing the total supply of a cryptocurrency or token. This is usually done to increase the value of the remaining tokens, as the reduced supply can create scarcity and potentially drive up demand.

Contract owner cannot burn tokens	✅ **The owner cannot burn tokens.**
Description	The owner is not able burn tokens without any allowances.
Comment	N/A

Blacklist addresses

Blacklisting addresses in smart contracts is the process of adding a certain address to a blacklist, effectively preventing them from accessing or participating in certain functionalities or transactions within the contract. This can be useful in preventing fraudulent or malicious activities, such as hacking attempts or money laundering.

Contract owner can blacklist addresses.	❌ **The owner can blacklist wallets.**
Description	The owner can blacklist wallets from transferring of tokens.
Comment	The contract contains the functionality in which the owner of the contract can blacklist the wallets from claiming the tokens from the contract, as this can potentially lead to the lock of funds for the user if the owner has blocked the user for an indefinite period of time.

File/Line(s): L417-421
Codebase: PresaleV2.sol

```
ftrace | funcSig
function blacklistUsers(address[] calldata _usersToBlacklist↑) external onlyOwner {
    for (uint256 i = 0; i < _usersToBlacklist↑.length; i++) {
        isBlacklisted[_usersToBlacklist↑[i]] = true;
    }
}
```

Fees and Tax

In some smart contracts, the owner or creator of the contract can set fees for certain actions or operations within the contract. These fees can be used to cover the cost of running the contract, such as paying for gas fees or compensating the contract's owner for their time and effort indeveloping and maintaining the contract.

Contract owner cannot levy high taxes	✅ **The owner cannot set fees more than 25%.**
Description	The owner cannot set fees of more than 25%.
Comment	N/A

Lock User Funds

In a smart contract, locking refers to the process of restricting access to certain tokens or assets for a specified period of time. When token or assets are locked in a smart contract, they cannot be transferred or used until the lock-up period has expired or certain conditions have been met.

Contract owner can lock function.	❌ **The owner can lock function.**
Description	The owner can lock the contract.
Comment	The contract imports the pausable contract through which the owner of the contract can pause and un-pause the claimed functionality in the contract for an indefinite period of time, which is not recommended as this can lock the tokens for the user, and the user will not be able to claim their tokens.

File/Line(s): L97-99

Codebase: PresaleV2.sol

```
function pause() external onlyOwner {
    _pause();
}
```

Centralization Privileges

Centralization can arise when one or more parties have privileged access or control over the contract's functionality, data, or decision-making. This can occur, for example,if the contract is controlled by a single entity or if certain participants have special permissions or abilities that others do not.

In the project, there are authorities that have access to the following functions:

File	Privileges
PresaleV2.sol	➢ The owner can pause/un-pause the contract buying and claim functionality for an indefinite period of time. ➢ The owner can change the sale start time only when the sale is not started and ended. ➢ The owner can change the wallet holding percentage. ➢ The owner can update the staking manager contract address. ➢ The owner can change the sale token and start the claim for whitelisted users only. Also, the owner can change the staking manager contract address. ➢ The owner can enable/disable the staking and claim for whitelisted users. ➢ The owner can enable/disable claiming. ➢ The owner can whitelist/un-whitelist users for the WERT token. ➢ The owner can update the max token buy amount to any arbitrary amount, excluding zero. ➢ The owner can blacklist users from transferring tokens. ➢ The owner can remove the users from blacklisted users. ➢ The owner can add/remove the user from the whitelisted list. ➢ The owner can change the payment wallet address. ➢ The owner can update any arbitrary value in the time constant, including zero. ➢ The owner can add users with other chains in the contract with the amount that they have deposited.

Recommendations

To avoid potential hacking risks, it is advisable for the client to manage the private key of the privileged account with care. Additionally, we

recommend enhancing the security practices of centralized privileges or roles in the protocol through a decentralized mechanism or smart-contract-based accounts, such as multi-signature wallets.

Here are some suggestions of what the client can do:

- Consider using multi-signature wallets: Multi-signature wallets require multiple parties to sign off on a transaction before it can be executed, providing an extra layer of security e.g. Gnosis Safe
- Use of a timelock at least with a latency of e.g. 48-72 hours for awareness of privileged operations
- Introduce a DAO/Governance/Voting module to increase transparency and user involvement
- Consider Renouncing the ownership so that the owner cannot modify any state variables of the contract anymore. Make sure to set up everything before renouncing.

Audit Result

Critical Issues

No critical issues

High Issues

No high issues

Medium Issue

#1 | The owner can lock rewards and tokens.

File	Severity	Location	Status
PresaleV2.sol	Medium	L346-349	Open

Description— The contract contains the functionality in which the owner can enable/disable the claimed functionality manually. This can lock the tokens in the contract for an indefinite period of time, which is not recommended, as it can lock the claim functionality, and user tokens will get stuck. There must be a check-in that the owner cannot disable the claim once enabled. Also, the owner can pause/un-pause the claim and other functionality for an indefinite period, which is not recommended.

#2 | Missing 'isContract' check.

File	Severity	Location	Status
PresaleV2.sol	Medium	L314-318	Open

Description – The owner of the contract can update the staking manager contract address to any arbitrary address excluding zero address. There must be a check so that this address can only be set to contract address to avoid any failure in the functionality.

#3 | Missing 'require' check (Potential honeypot).

File	Severity	Location	Status
PresaleV2.sol	Medium	L465-468	Open

Description – The owner can set any arbitrary address to the payment wallet address, excluding zero address as this can lead to a potential honeypot if the owner has set the address to a contract address that cannot receive ETH. It is recommended that the address cannot be set to a contract address that cannot receive ETH to avoid these circumstances.

#4 | Missing 'require' check.

File	Severity	Location	Status
PresaleV2.sol	Medium	L474-476	Open

Description – The owner of the contract can update any value in the time constant, including zero this can fail the buy functionality in the contract it is recommended that there should be a minimum time constant of 1 day to avoid these scenarios in the contract.

#5 | The owner can blacklist wallets from claiming.

File	Severity	Location	Status
PresaleV2.sol	Medium	L417-421	Open

Description – The contract contains the functionality in which the owner of the contract can blacklist the wallets from claiming the tokens from the contract, as this can potentially lead to the lock of funds for the user if the owner has blocked the user for an indefinite period of time.

Low Issue

#1 | Missing events

File	Severity	Location	Status
TheClubPresale.sol	Low	L457-459, L465-468, L474-476, L338-340, L346-349	Open

Description – It is recommended to emit all the critical parameter changes in the contract.

#2 | Floating pragma solidity version.

File	Severity	Location	Status
PresaleV2.sol	Low	L10	Open

Description – Adding the constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

Informational Issue

No informational issues

Legend for the Issue Status

Attribute or Symbol	Meaning
Open	The issue is not fixed by the project team.
Fixed	The issue is fixed by the project team.
Acknowledged(ACK)	The issue has been acknowledged or declared as part of business logic.

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY