

KISHIELD

Security Audit

DOGPET

6/26/2024



Audit Passed

Risk Analysis

The table below provides a comprehensive summary of the project's risk analysis, emphasizing high-risk functions that could potentially transform the contract into a honeypot.

This analysis is derived from the critical insights of our thorough audit, focusing on the implications and impact of these findings on the overall integrity and security of the project.

Cat.	Detector	Description
	Can mint	The contract cannot mint tokens
	Tax above 25%	Tax is below 25%
	Max tx	There is no max TX or has reasonable limits
	Min/Max wallet	There is no wallet limits or has reasonable limits
	Enable trade	Trade is already enabled
	Modify tax	Tax cannot be modified or less than 25% total
	Can blacklist	The contract cannot blacklist wallets
	Is honeypot	The contract is not a honeypot
	Trading cooldown	There is a cooldown, risk of not being able to sell
	Pause trade	Trade is paused, risk of not being able to sell
	Pause transfer	Transfer cannot be paused
	Is proxy	The contract is not a proxy

Table of Contents

- 1 Audit Summary**
- 2 Project Overview**
 - 2.1 Token Summary
 - 2.2 Main Contract Assessed
- 3 Smart Contract Vulnerability Checks**
- 4 Contract Ownership**
 - 4.1 Privileged Functions
- 5 Important Notes To The Users**
- 6 Findings Summary**
 - 6.1 Classification of Issues
 - 6.1 Findings Table
 - 01 Divide before multiply
 - 02 Reentrancy vulnerabilities
 - 03 Local variable shadowing
 - 04 Block timestamp
 - 05 Dead-code
 - 06 Incorrect versions of Solidity
 - 07 Too many digits
 - 08 State variables that could be declared constant
 - 09 State variables that could be declared immutable
- 7 Statistics**
 - 7.1 Liquidity
 - 7.2 Token Holders
 - 7.3 Liquidity Holders

8 Liquidity Ownership

9 Disclaimer

Audit Summary

This report has been prepared for **DOGPET** on the Binance Smart Chain network. **KISHIELD** provides both client-centered and user-centered examination of the smart contracts and their current status when applicable. This report represents the security assessment made to find issues and vulnerabilities on the source code along with the current liquidity and token holder statistics of the protocol.

A comprehensive examination has been performed, utilizing Cross Referencing, Static Analysis, In-House Security Tools, and line-by-line Manual Review.

The auditing process pays special attention to the following considerations:

- Ensuring contract logic meets the specifications and intentions of the client without exposing the user's funds to risk.
- Testing the smart contracts against both common and uncommon attack vectors.
- Inspecting liquidity and holders statistics to inform the current status to both users and client when applicable.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Verifying contract functions that allow trusted and/or untrusted actors to mint, lock, pause, and transfer assets.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Project Overview

Token Summary

Parameter	Result
Address	0x21dfe97101717ed7f562da5d1ccbceef8fef33c3
Name	DOGPET
Token Tracker	DOGPET (DP)
Decimals	18
Supply	21000000000
Platform	Binance Smart Chain
compiler	v0.8.19+commit.7dd6d404
Optimization	Yes with 1 runs
LicenseType	SPDX-License-Identifier: MIT
Language	Solidity
Codebase	56
Url	https://dogepet.world

Main Contract Assessed

Name	Contract	Live
DOGPET	0x21dfe97101717ed7f562da5d1ccbceef8fef33c3	Yes

Statistics

Liquidity Info

Parameter	Result
Pair Address	0x1cdefd394a8144381ea1626bd51bb81818686a77
DP Reserves	283559164.10028136 DP
BNB Reserves	95418.33112200466 BNB
Liquidity Value	\$94201 USD

Token (DP) Holders Info

Parameter	Result
DP Percentage Locked	0.75%
DP Amount Locked	15,794,836,507 DP
Top 10 Percentage Own	0.88%
Top 10 Amount Owned	18,377,844,619 DP

LP (DP/BNB) Holders Info

No LP holders found

* All the data displayed above was taken on-chain on 2024/06/26

**Data is delivered as is, we do not take responsibility for any errors or omissions in the data

Important Notes To The Users:

- General Information
- Max Transfer: No single transfer can exceed 50% of the liquidity pool's balance.
- Sell Fee: Selling DogPet tokens incurs a 6% fee.
- Buy Fee: Buying DogPet tokens incurs a 6% fee.
- Transfer Fee: All token transfers include a 2% fee for the foundation and market.
- Important Information
- Trading Cooldown: After a significant sell transaction, trading is paused for 10 minutes to stabilize the price. During this period, the foundationAddress works to balance the price.
- Privileged Addresses: Certain addresses, such as the foundationAddress and destroyAddress, have special permissions and can perform essential operations without restrictions.
- Price Stability: The trading cooldown mechanism and privileged addresses ensure the token's price remains stable and prevents drastic fluctuations.

Read carefully the notes section and make your own decision before interacting with the audited contract.

Audit Passed



Smart Contract Vulnerability Checks

Vulnerability	Automatic Scan	Manual Scan	Result
Unencrypted Private Data On-Chain	Complete	Complete	✓ Low / No Risk
Code With No Effects	Complete	Complete	✓ Low / No Risk
Message call with hardcoded gas amount	Complete	Complete	✓ Low / No Risk
Hash Collisions With Multiple Variable Length Arguments	Complete	Complete	✓ Low / No Risk
Unexpected Ether balance	Complete	Complete	✓ Low / No Risk
Presence of unused variables	Complete	Complete	✓ Low / No Risk
Right-To-Left-Override control character (U+202E)	Complete	Complete	✓ Low / No Risk
Typographical Error	Complete	Complete	✓ Low / No Risk
DoS With Block Gas Limit	Complete	Complete	✓ Low / No Risk
Arbitrary Jump with Function Type Variable	Complete	Complete	✓ Low / No Risk
Insufficient Gas Griefing	Complete	Complete	✓ Low / No Risk
Incorrect Inheritance Order	Complete	Complete	✓ Low / No Risk
Write to Arbitrary Storage Location	Complete	Complete	✓ Low / No Risk
Requirement Violation	Complete	Complete	✓ Low / No Risk
Missing Protection against Signature Replay Attacks	Complete	Complete	✓ Low / No Risk
Weak Sources of Randomness from Chain Attributes	Complete	Complete	✓ Low / No Risk

Vulnerability	Automatic Scan	Manual Scan	Result
Authorization through tx.origin	Complete	Complete	✓ Low / No Risk
Delegatecall to Untrusted Callee	Complete	Complete	✓ Low / No Risk
Use of Deprecated Solidity Functions	Complete	Complete	✓ Low / No Risk
Assert Violation	Complete	Complete	✓ Low / No Risk
Reentrancy	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
Unprotected Ether Withdrawal	Complete	Complete	✓ Low / No Risk
Unchecked Call Return Value	Complete	Complete	✓ Low / No Risk
Outdated Compiler Version	Complete	Complete	✓ Low / No Risk
Integer Overflow and Underflow	Complete	Complete	✓ Low / No Risk
Function Default Visibility	Complete	Complete	✓ Low / No Risk

Contract Ownership

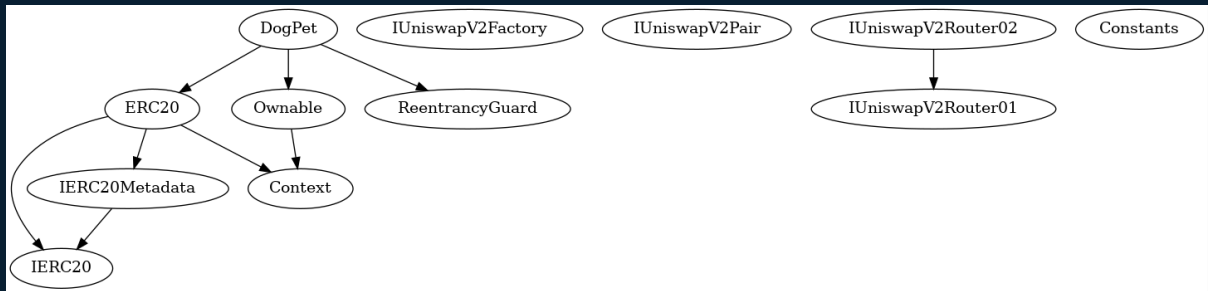
The contract does not have an owner.

Having no owner means that all the ownable functions in the contract can not be called by anyone, this often leads to more trust on the project.

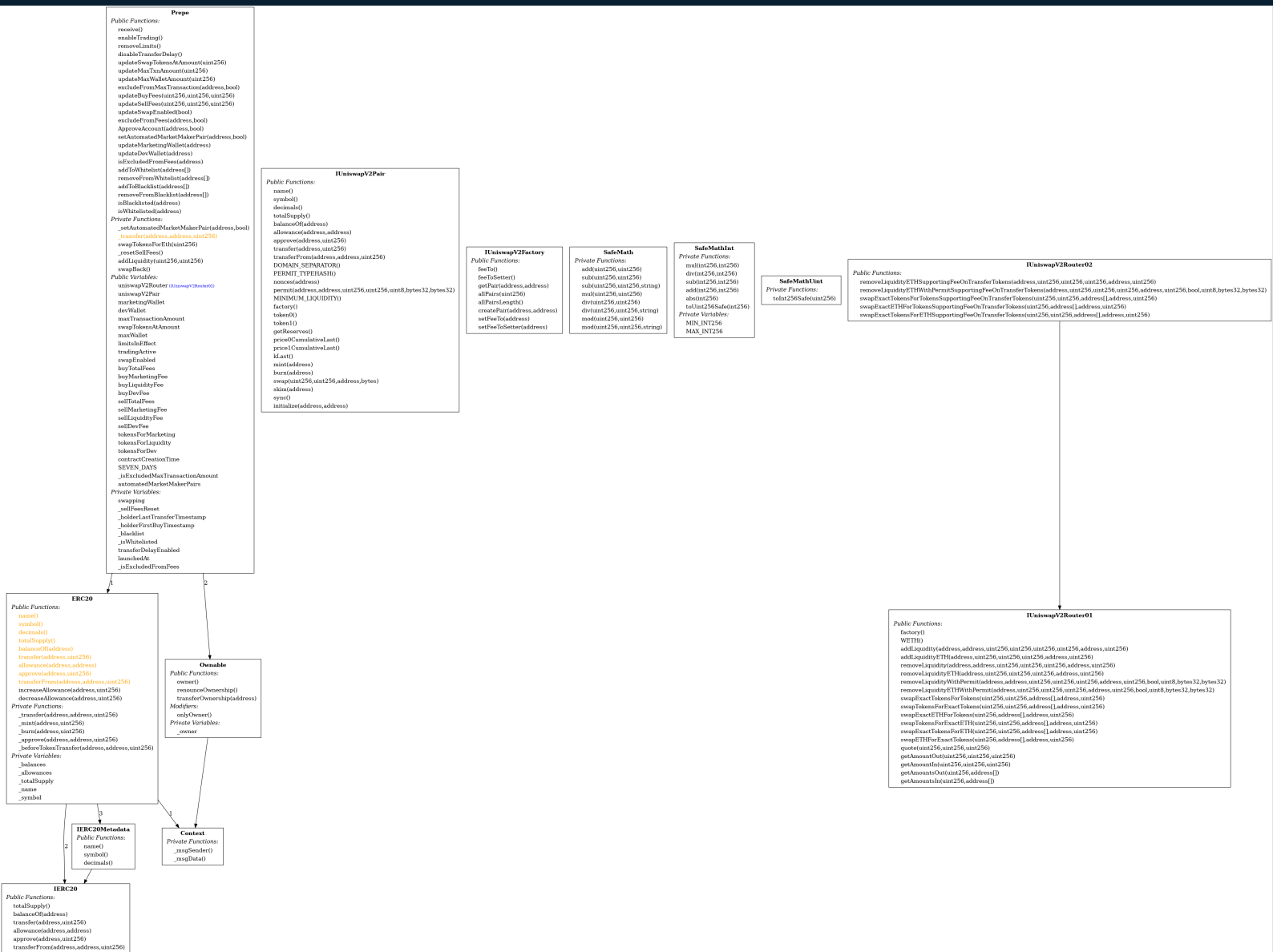
This contract has a PairManager that can call special functions:
0xe7b80ad1a73cec99eac142ccb4a6e6a913743cea



Inheritance Graph For DOGPET







Full Inheritance Graph For DOGPET







Technical Findings Summary

Classification of Issues

*All Issues Found are Informational

Severity	Description
 High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency
 Medium	Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
 Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
 Info	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

Findings

Severity	Found
 High	0
 Medium	2
 Low	2
 Info	3
Total	7

Findings

Divide before multiply

ID	Severity	Detector	Description
01	Medium	divide-before-multiply	Solidity's integer division truncates. Thus, performing division before multiplication can lead to precision loss.

Description

DogPet._swapAndTransferFees(address,uint256,bool) (contracts/AuditContract.sol#1130-1148) performs a multiplication on the result of a division:

```
- fee = (amount * _FEE) / 100 (contracts/AuditContract.sol#1135)
  - super._transfer(from,address(this),fee * 2) (contracts/AuditContract.sol#1139)
```

Exploit Example

```
contract A {
  function f(uint n) public {
    coins = (oldSupply / n) * interest;
  }
}
```

Recommendation

Consider ordering multiplication before division.

Reentrancy vulnerabilities

ID	Severity	Detector	Description
02	Medium	reentrancy-no-eth	Detection of the [reentrancy bug](https://github.com/trailofbits/not-so-smart-contracts/tree/master/reentrancy). Do not report reentrancies that involve Ether (see `reentrancy-eth`).

Description

Reentrancy in DogPet._handleBuy(address,address,uint256) (contracts/AuditContract.sol#1124-1128):

External calls:

- _swapAndTransferFees(from,amount,false) (contracts/AuditContract.sol#1125)
- router.swapExactTokensForTokensSupportingFeeOnTransferTokens(amount,minOutputAmount,path,to,block.timestamp) (contracts/AuditContract.sol#1159-1165)

State variables written after the call(s):

- super._transfer(from,to,buyAmount) (contracts/AuditContract.sol#1127)
- _balances[from] = fromBalance - amount (contracts/AuditContract.sol#387)
- _balances[to] += amount (contracts/AuditContract.sol#390)

ERC20._balances (contracts/AuditContract.sol#164) can be used in cross function reentrancies:

- ERC20._mint(address,uint256) (contracts/AuditContract.sol#407-420)
- ERC20._transfer(address,address,uint256) (contracts/AuditContract.sol#371-396)
- ERC20.balanceOf(address) (contracts/AuditContract.sol#226-230)

Reentrancy in DogPet._handleSell(address,address,uint256) (contracts/AuditContract.sol#1118-1122):

External calls:

- _swapAndTransferFees(from,amount,true) (contracts/AuditContract.sol#1119)
- router.swapExactTokensForTokensSupportingFeeOnTransferTokens(amount,minOutputAmount,path,to,block.timestamp) (contracts/AuditContract.sol#1159-1165)

State variables written after the call(s):

- super._transfer(from,to,sellAmount) (contracts/AuditContract.sol#1121)
- _balances[from] = fromBalance - amount (contracts/AuditContract.sol#387)
- _balances[to] += amount (contracts/AuditContract.sol#390)

ERC20._balances (contracts/AuditContract.sol#164) can be used in cross function reentrancies:

- ERC20._mint(address,uint256) (contracts/AuditContract.sol#407-420)
- ERC20._transfer(address,address,uint256) (contracts/AuditContract.sol#371-396)
- ERC20.balanceOf(address) (contracts/AuditContract.sol#226-230)

Reentrancy in DogPet._swapAndTransferFees(address,uint256,bool) (contracts/AuditContract.sol#1130-1148):

External calls:

- _sellSwapToWeight(lastFoundationAmount,foundationAddress) (contracts/AuditContract.sol#1143)
- router.swapExactTokensForTokensSupportingFeeOnTransferTokens(amount,minOutputAmount,path,to,block.timestamp) (contracts/AuditContract.sol#1159-1165)
- _sellSwapToWeight(lastMarketAmount,marketAddress) (contracts/AuditContract.sol#1144)
- router.swapExactTokensForTokensSupportingFeeOnTransferTokens(amount,minOutputAmount,path,to,block.timestamp) (contracts/AuditContract.sol#1159-1165)

State variables written after the call(s):

- lastFoundationAmount = 0 (contracts/AuditContract.sol#1145)

DogPet.lastFoundationAmount (contracts/AuditContract.sol#1013) can be used in cross function reentrancies:

- DogPet._swapAndTransferFees(address,uint256,bool) (contracts/AuditContract.sol#1130-1148)
- lastMarketAmount = 0 (contracts/AuditContract.sol#1146)

DogPet.lastMarketAmount (contracts/AuditContract.sol#1014) can be used in cross function reentrancies:

- DogPet._swapAndTransferFees(address,uint256,bool) (contracts/AuditContract.sol#1130-1148)

Exploit Example

```
function bug(){
    require(not_called);
    if( ! (msg.sender.call() ) ){
        throw;
    }
    not_called = False;
}
```

Recommendation

Apply the [check-effects-interactions` pattern](<http://solidity.readthedocs.io/en/v0.4.21/security-considerations.html#re-entrancy>).

Local variable shadowing

ID	Severity	Detector	Description
03	Low	shadowing-local	Detection of shadowing using local variables.

Description

DogPet.constructor().totalSupply (contracts/AuditContract.sol#1031) shadows:
- ERC20.totalSupply() (contracts/AuditContract.sol#219-221) (function)
- IERC20.totalSupply() (contracts/AuditContract.sol#29) (function)

Exploit Example

```
pragma solidity ^0.4.24;

contract Bug {
    uint owner;

    function sensitive_function(address owner) public {
        // ...
        require(owner == msg.sender);
    }

    function alternate_sensitive_function() public {
        address owner = msg.sender;
        // ...
        require(owner == msg.sender);
    }
}
```

Recommendation

Rename the local variables that shadow another component.

Block timestamp

ID	Severity	Detector	Description
04	Low	timestamp	Dangerous usage of `block.timestamp`. `block.timestamp` can be manipulated by miners.

Description

DogPet._destroyTokens() (contracts/AuditContract.sol#1170-1185) uses timestamp for comparisons
Dangerous comparisons:

- block.timestamp >= lastRecordedTime (contracts/AuditContract.sol#1172)

DogPet._transfer(address,address,uint256) (contracts/AuditContract.sol#1065-1116) uses timestamp for comparisons
Dangerous comparisons:

- require(bool,string)(block.timestamp >= sellTime || !_isAddLiquidityAddress[from] || sellResume,Sell: Trading is temporarily disabled.) (contracts/AuditContract.sol#1096-1101)

Exploit Example

"Bob's contract relies on `block.timestamp` for its randomness. Eve is a miner and manipulates `block.timestamp` to exploit Bob's contract.

Recommendation

Avoid relying on `block.timestamp`.

Dead-code

ID	Severity	Detector	Description
05	Informational	dead-code	Functions that are not used.

Description

Context._contextSuffixLength() (contracts/AuditContract.sol#130-132) is never used and should be removed

ERC20._burn(address,uint256) (contracts/AuditContract.sol#433-449) is never used and should be removed

Context._msgData() (contracts/AuditContract.sol#126-128) is never used and should be removed

ReentrancyGuard._reentrancyGuardEntered() (contracts/AuditContract.sol#976-978) is never used and should be removed

Exploit Example

```
contract Contract{
    function dead_code() internal() {}
}
```

Recommendation

Remove unused functions.

Incorrect versions of Solidity

ID	Severity	Detector	Description
06	Informational	solc-version	<code>`solc`</code> frequently releases new compiler versions. Using an old version prevents access to new Solidity security checks. We also recommend avoiding complex <code>`pragma`</code> statement.

Description

Version constraint 0.8.9 contains known severe issues (<https://solidity.readthedocs.io/en/latest/bugs.html>)

- VerbatimInvalidDeduplication
 - FullInlinerNonExpressionSplitArgumentEvaluationOrder
 - MissingSideEffectsOnSelectorAccess
 - AbiReencodingHeadOverflowWithStaticArrayCleanup
 - DirtyByteArrayToStorage
 - DataLocationChangeInInternalOverride
 - NestedCalldataArrayAbiReencodingSizeValidation.

It is used by:

- 0.8.9 (contracts/AuditContract.sol#2)

Exploit Example

Recommendation

Deploy with any of the following Solidity versions:

- 0.8.18

The recommendations take into account:

- Risks related to recent releases
- Risks of complex code generation changes
- Risks of new language features
- Risks of known bugs

Use a simple pragma version that allows any of these versions.
Consider using the latest version of Solidity for testing.

Too many digits

ID	Severity	Detector	Description
07	Informational	too-many-digits	Literals with many digits are difficult to read and review.

Description

DogPet.constructor() (contracts/AuditContract.sol#1030-1051) uses literals with too many digits:
- totalSupply = 2100000000000000000000000000 (contracts/AuditContract.sol#1031)

[illegible]

Exploit Example

```
contract MyContract{
    uint 1_ether = 1000000000000000000;
}
```

Recommendation

Use:

- [Ether suffix](https://solidity.readthedocs.io/en/latest/units-and-global-variables.html#ether-units),
- [Time suffix](https://solidity.readthedocs.io/en/latest/units-and-global-variables.html#time-units), or
- [The scientific notation](https://solidity.readthedocs.io/en/latest/types.html#rational-and-integer-literals)

State variables that could be declared constant

ID	Severity	Detector	Description
08	Optimization	constable-states	State variables that are not updated following deployment should be declared constant to save gas.

Description

DogPet.gameMitAddress (contracts/AuditContract.sol#1009) should be constant

DogPet.dividendAddress (contracts/AuditContract.sol#1004) should be constant

DogPet.destroyAddress (contracts/AuditContract.sol#1003) should be constant

DogPet.marketAddress (contracts/AuditContract.sol#1008) should be constant

DogPet.forPinkSaleAddress (contracts/AuditContract.sol#1010-1011) should be constant

DogPet.foundationAddress (contracts/AuditContract.sol#1006-1007) should be constant

Exploit Example

Recommendation

Add the `constant` attribute to state variables that never change.

State variables that could be declared immutable

ID	Severity	Detector	Description
09	Optimization	immutable-states	State variables that are not updated following deployment should be declared immutable to save gas.

Description

DogPet.pair (contracts/AuditContract.sol#993) should be immutable

DogPet.router (contracts/AuditContract.sol#992) should be immutable

Exploit Example

Recommendation

Add the `immutable` attribute to state variables that never change or are set only in the constructor.

Privileged Functions (onlyOwner & Others)

Function Name	Parameters
renounceOwnership	onlyOwner
transferOwnership	onlyOwner
transfer	nonReentrant
transferFrom	nonReentrant
setAddLiquidityAddress	onlyOwner
_transfer	nonReentrant
_handleSell	nonReentrant
_handleBuy	nonReentrant
_swapAndTransferFees	nonReentrant

Disclaimer

KISHIELD has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocacy for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and KISHIELD is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will KISHIELD or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

The assessment services provided by KISHIELD 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.