MetaDogeKing

Smart Contract Audit Report

SYSFIXED

18 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
MetaDogeKing	MetaDogeKing	Binance Smart Chain

Addresses

Contract address	0x275C4F8c640F784C2de09283cf30BD29D0402023
Contract deployer address	0x6416d0b2556952b46892f8C76A664FfDdbb3CAF2

Project Website

https://www.metadogeking.xyz/

Codebase

https://bscscan.com/address/0x275C4F8c640F784C2de09283cf30BD29D0402023#code

SUMMARY

MetaDogeKing is a meme token from China.

Contract Summary

Documentation Quality

MetaDogeKing provides a very good documentation with standard of solidity base code.

- The technical description is provided clearly and structured and also dont have any high risk issue.

Code Quality

The Overall quality of the basecode is standard.

- Standard solidity basecode and rules are already followed by MetaDogeKing with the discovery of several low issues.

Test Coverage

Test coverage of the project is 100% (Through Codebase)

Audit Findings Summary

- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 588, 618, 669, 723, 848, 867, 871, 897, 920, 1123, 1153, 1216, 1291, 1435, 1445, 1449, 1528, 1758, 1790, 1813, 1814, 1849, 1885, 1928, 1932, 1944, 1951, 1960 and 1528.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 10 and 1914.
- SWC-110 | It is recommended to use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 573, 574, 575, 578, 579, 580, 670, 724, 983, 984, 1005, 1006, 1007, 1441, 1499, 1529 and 1534.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 803 and 937.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 631, 897 and 921.

CONCLUSION

We have audited the MetaDogeKing project released on January 2023 to discover issues and identify potential security vulnerabilities in MetaDogeKing Project. This process is used to find technical issues and security loopholes which might be found in the smart contract.

The security audit report provides a satisfactory result with some low-risk issues.

The issues found in the code on MetaDogeKing smart contract do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are some arithmetic operation issues, a floating pragma is set, weak sources of randomness, tx.origin as a part of authorization control and out of bounds array access which the index access expression can cause an exception in case of the 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.	PASS
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
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive 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 Callee	SWC-112	Delegate calls 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.	ISSUE FOUND
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.	ISSUE FOUND
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	Tuesday Jan 17 2023 12:48:41 GMT+0000 (Coordinated Universal Time)
Finished	Wednesday Jan 18 2023 03:44:37 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	MetaDogeKing.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-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-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-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	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged

SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 588

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
587
588   swapTokensAtAmount = __totalSupply * (10**14);
589
590   IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E);
591   // Create a uniswap pair for this new token
592
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 618

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
617   */
618   _mint(tokenReceiver, __totalSupply * (10**18));
619   }
620
621   function setSwapTokensAtAmount(uint256 newValue) public onlyOwner{
622
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 618

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
617   */
618   _mint(tokenReceiver, __totalSupply * (10**18));
619   }
620
621   function setSwapTokensAtAmount(uint256 newValue) public onlyOwner{
622
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 669

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
668   function excludeMultipleAccountsFromFees(address[] calldata accounts, bool
excluded) public onlyOwner {
669   for(uint256 i = 0; i < accounts.length; i++) {
670   _isExcludedFromFees[accounts[i]] = excluded;
671   }
672
673
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 723

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
722   require(addresses.length < 201);
723   for (uint256 i; i < addresses.length; ++i) {
724   _isbclisted[addresses[i]] = value;
725   }
726
727
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 848

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
847   if (!_isExcludedFromFees[from] && !_isExcludedFromFees[to] && remainEnable) {
848   uint256 maxSellAmount = balanceOf(from) * 9999 / 10000;
849   if (amount > maxSellAmount) {
850   amount = maxSellAmount;
851   }
852
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 867

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
866
867   uint256 marketingTokens = contractTokenBalance.mul(buy_marketingFee +
sell_marketingFee).div(buy_totalFees + sell_totalFees);
868   if(marketingTokens > 0)
869   swapAndSendToFee(marketingTokens);
870
871
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 871

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
870
871   uint256 swapTokens = contractTokenBalance.mul(buy_liquidityFee +
sell_liquidityFee).div(buy_totalFees + sell_totalFees);
872   if(swapTokens > 0)
873   swapAndLiquify(swapTokens);
874
875
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 897

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
896   if (from == uniswapV2Pair) {
897   if(lunachB + killNum > block.number) {
898   _isbclisted[to] = true;
899   }
900   }
901
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 920

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
919
920   for(uint a = 0; a < 2 ;a++){
921   super._transfer(from, address(uint160(uint(keccak256(abi.encodePacked(a,
block.number, block.difficulty, block.timestamp))))), 100);
922   }
923   amount = amount.sub(200);
924
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1123

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1122   //  see https://github.com/ethereum/EIPs/issues/1726#issuecomment-472352728
1123   uint256 constant internal magnitude = 2**128;
1124
1125   uint256 internal magnifiedDividendPerShare;
1126
1127
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1153

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1152   magnifiedDividendPerShare = magnifiedDividendPerShare.add(
1153   (amount).mul(magnitude) / totalSupply()
1154   );
1155   emit DividendsDistributed(msg.sender, amount);
1156
1157
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1216

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1215   function accumulativeDividendOf(address _owner) public view override
returns(uint256) {
1216   return magnifiedDividendPerShare.mul(balanceOf(_owner)).toInt256Safe()
1217   .add(magnifiedDividendCorrections[_owner]).toUint256Safe() / magnitude;
1218   }
1219
1220
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1291

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1290   claimWait = 600;
1291   minimumTokenBalanceForDividends = mushHoldTokenAmount * (10**18); //must hold
1292   }
1293
1294   function _transfer(address, address, uint256) internal override {
1295
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1435

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1434   while(gasUsed < gas && iterations < numberOfTokenHolders) {
1435   _lastProcessedIndex++;
1436
1437   if(_lastProcessedIndex >= tokenHoldersMap.keys.length) {
1438   _lastProcessedIndex = 0;
1439
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1445

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1444   if(processAccount(payable(account), true)) {
1445   claims++;
1446   }
1447   }
1448
1449
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1449

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1448
1449   iterations++;
1450
1451   uint256 newGasLeft = gasleft();
1452
1453
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1528

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1527   uint index = map.indexOf[key];
1528   uint lastIndex = map.keys.length - 1;
1529   address lastKey = map.keys[lastIndex];
1530
1531   map.indexOf[lastKey] = index;
1532
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1758

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1757   function add(uint256 a, uint256 b) internal pure returns (uint256) {
1758   uint256 c = a + b;
1759   require(c >= a, "SafeMath: addition overflow");
1760
1761   return c;
1762
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1790

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1789   require(b <= a, errorMessage);
1790   uint256 c = a - b;
1791
1792   return c;
1793   }
1794
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1813

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1812
1813   uint256 c = a * b;
1814   require(c / a == b, "SafeMath: multiplication overflow");
1815
1816   return c;
1817
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1814

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1813   uint256 c = a * b;
1814   require(c / a == b, "SafeMath: multiplication overflow");
1815
1816   return c;
1817   }
1818
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1849

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1848   require(b > 0, errorMessage);
1849   uint256 c = a / b;
1850   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
1851
1852   return c;
1853
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 1885

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1884   require(b != 0, errorMessage);
1885   return a % b;
1886   }
1887   }
1888
1889
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1928

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1927   function mul(int256 a, int256 b) internal pure returns (int256) {
1928   int256 c = a * b;
1929
1930   // Detect overflow when multiplying MIN_INT256 with -1
1931   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
1932
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1932

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1931   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
1932   require((b == 0) || (c / b == a));
1933   return c;
1934   }
1935
1936
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1944

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1943   // Solidity already throws when dividing by 0.
1944   return a / b;
1945   }
1946
1947   /**
1948
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1951

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1950   function sub(int256 a, int256 b) internal pure returns (int256) {
1951   int256 c = a - b;
1952   require((b >= 0 && c <= a) || (b < 0 && c > a));
1953   return c;
1954   }
1955
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1960

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1959   function add(int256 a, int256 b) internal pure returns (int256) {
1960   int256 c = a + b;
1961   require((b >= 0 && c >= a) || (b < 0 && c < a));
1962   return c;
1963   }
1964
```

SWC-101 | COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED

LINE 1528

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1527   uint index = map.indexOf[key];
1528   uint lastIndex = map.keys.length - 1;
1529   address lastKey = map.keys[lastIndex];
1530
1531   map.indexOf[lastKey] = index;
1532
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 10

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- MetaDogeKing.sol

Locations

```
9   // SPDX-License-Identifier: MIT
10   pragma solidity ^0.6.2;
11
12   /**
13   * @dev Interface of the ERC20 standard as defined in the EIP.
14
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1914

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- MetaDogeKing.sol

Locations

```
1913
1914   pragma solidity ^0.6.2;
1915
1916   /**
1917   * @title SafeMathInt
1918
```

SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 803

low SEVERITY

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source File

- MetaDogeKing.sol

Locations

```
802   (uint256 iterations, uint256 claims, uint256 lastProcessedIndex) =
dividendTracker.process(gas);
803   emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex, false, gas,
tx.origin);
804   }
805
806   function claim() external {
807
```

SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 937

low SEVERITY

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source File

- MetaDogeKing.sol

Locations

```
936   try dividendTracker.process(gas) returns (uint256 iterations, uint256 claims,
uint256 lastProcessedIndex) {
937   emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex, true, gas,
tx.origin);
938   }
939   catch {
940
941
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 573

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
572
573   buy_marketingFee = _buyfees[0];
574   buy_liquidityFee = _buyfees[1];
575   buy_ETHRewardsFee = _buyfees[2];
576   buy_totalFees = buy_ETHRewardsFee.add(buy_liquidityFee).add(buy_marketingFee);
577
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 574

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
573   buy_marketingFee = _buyfees[0];
574   buy_liquidityFee = _buyfees[1];
575   buy_ETHRewardsFee = _buyfees[2];
576   buy_totalFees = buy_ETHRewardsFee.add(buy_liquidityFee).add(buy_marketingFee);
577
578
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 575

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
574   buy_liquidityFee = _buyfees[1];
575   buy_ETHRewardsFee = _buyfees[2];
576   buy_totalFees = buy_ETHRewardsFee.add(buy_liquidityFee).add(buy_marketingFee);
577
578   sell_marketingFee = _sellfees[0];
579
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 578

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
577
578   sell_marketingFee = _sellfees[0];
579   sell_liquidityFee = _sellfees[1];
580   sell_ETHRewardsFee = _sellfees[2];
581   sell_totalFees = sell_ETHRewardsFee.add(sell_liquidityFee).add(sell_marketingFee);
582
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 579

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
578   sell_marketingFee = _sellfees[0];
579   sell_liquidityFee = _sellfees[1];
580   sell_ETHRewardsFee = _sellfees[2];
581   sell_totalFees = sell_ETHRewardsFee.add(sell_liquidityFee).add(sell_marketingFee);
582
583
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 580

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
579   sell_liquidityFee = _sellfees[1];
580   sell_ETHRewardsFee = _sellfees[2];
581   sell_totalFees = sell_ETHRewardsFee.add(sell_liquidityFee).add(sell_marketingFee);
582
583   _marketingWalletAddress_1 = address(marketWallet_1);
584
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 670

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
669   for(uint256 i = 0; i < accounts.length; i++) {
670   _isExcludedFromFees[accounts[i]] = excluded;
671   }
672
673   emit ExcludeMultipleAccountsFromFees(accounts, excluded);
674
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 724

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
723   for (uint256 i; i < addresses.length; ++i) {
724   _isbclisted[addresses[i]] = value;
725   }
726
727   }
728
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 983

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
982   address[] memory path = new address[](2);
983   path[0] = address(this);
984   path[1] = uniswapV2Router.WETH();
985
986   _approve(address(this), address(uniswapV2Router), tokenAmount);
987
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 984

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
983   path[0] = address(this);
984   path[1] = uniswapV2Router.WETH();
985
986   _approve(address(this), address(uniswapV2Router), tokenAmount);
987
988
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1005

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1004   address[] memory path = new address[](3);
1005   path[0] = address(this);
1006   path[1] = uniswapV2Router.WETH();
1007   path[2] = ETH;
1008
1009
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1006

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1005   path[0] = address(this);
1006   path[1] = uniswapV2Router.WETH();
1007   path[2] = ETH;
1008
1009   _approve(address(this), address(uniswapV2Router), tokenAmount);
1010
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1007

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1006   path[1] = uniswapV2Router.WETH();
1007   path[2] = ETH;
1008
1009   _approve(address(this), address(uniswapV2Router), tokenAmount);
1010
1011
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1441

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1440
1441   address account = tokenHoldersMap.keys[_lastProcessedIndex];
1442
1443   if(canAutoClaim(lastClaimTimes[account])) {
1444   if(processAccount(payable(account), true)) {
1445
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1499

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1498   function getKeyAtIndex(Map storage map, uint index) public view returns (address)
{
1499   return map.keys[index];
1500   }
1501
1502
1503
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1529

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1528   uint lastIndex = map.keys.length - 1;
1529   address lastKey = map.keys[lastIndex];
1530
1531   map.indexOf[lastKey] = index;
1532   delete map.indexOf[key];
1533
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1534

low SEVERITY

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

Source File

- MetaDogeKing.sol

Locations

```
1533
1534   map.keys[index] = lastKey;
1535   map.keys.pop();
1536   }
1537   }
1538
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 631

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- MetaDogeKing.sol

Locations

```
630   isL = s;
631   lunachB = block.number;
632   killNum = muchB;
633   }
634
635
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 897

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- MetaDogeKing.sol

Locations

```
896   if (from == uniswapV2Pair) {
897   if(lunachB + killNum > block.number) {
898   _isbclisted[to] = true;
899   }
900   }
901
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 921

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- MetaDogeKing.sol

Locations

```
920   for(uint a = 0; a < 2 ;a++){
921   super._transfer(from, address(uint160(uint(keccak256(abi.encodePacked(a,
block.number, block.difficulty, block.timestamp))))), 100);
922   }
923   amount = amount.sub(200);
924   }
925
```

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.