Optimus Inu

Smart Contract Audit Report

SYSFIXED

26 Sep 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Optimus Inu	OPINU	Binance Smart Chain

Addresses

Contract address	0x674aa28ac436834051fff3fc7b6e59d6f9c57a1c
Contract deployer address	0x04c6d39453855721A1b8C2b8386cFeBaF66853Ae

Project Website

https://optimusinu.io/

Codebase

https://bscscan.com/address/0x674aa28ac436834051fff3fc7b6e59d6f9c57a1c#code

SUMMARY

The Optimus Inu ecosystem will have many utilities for holders of the OPINU token to increase their exposure in decentralized finance. This allows members to grow with the project as Elon's new robot Optimus takes over the technology world. Robots and crypto are the future of the economy. Optimus is the future of Robots. Optimus Inu is the future of cryptocurrency. Just as Elon's dog Floki before it, Elon's robot Optimus will have support from the general public and from Elon's popularity on social media. Come join the experienced team of OPINU and get ready to go to Mars!

Contract Summary

Documentation Quality

Optimus Inu 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 Optimus Inu 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 89, 100, 106, 107, 117, 173, 173, 174, 174, 175, 175, 227, 227, 441, 441, 446, 446, 480, 481, 491, 544 and 544.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 5.
- SWC-110 SWC-123 | It is recommended to use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 397, 398 and 492.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 304.

CONCLUSION

We have audited the Optimus Inu project released on September 2022 to discover issues and identify potential security vulnerabilities in Optimus Inu 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 satisfactory results with low-risk issues.

The issues found in the Optimus Inu smart contract code 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, Potential use of "block.number" as a source of randomness, and out-of-bounds array access in which the index access expression can cause an exception in case an invalid array index value is used.

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
Unprotected Ether Withdrawal	SWC-105	Due to missing or insufficient access controls, malicious parties can withdraw from the contract.	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
Uninitialized Storage Pointer	SWC-109	Uninitialized local storage variables can point to unexpected storage locations in the contract.	PASS
Assert Violation	SWC-110 SWC-123	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	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
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	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
Write to Arbitrary Storage Location	SWC-124	The contract is responsible for ensuring that only authorized user or contract accounts may write to sensitive storage locations.	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
Insufficient Gas Griefing	SWC-126	Insufficient gas griefing attacks can be performed on contracts which accept data and use it in a sub-call on another contract.	PASS
Arbitrary Jump Function	SWC-127	As Solidity doesnt support pointer arithmetics, it is impossible to change such variable to an arbitrary value.	PASS

Typographical Error	SWC-129	A typographical error can occur for example when the intent of a defined operation is to sum a number to a variable.	PASS
Override control character	SWC-130	Malicious actors can use the Right-To-Left-Override unicode character to force RTL text rendering and confuse users as to the real intent of a contract.	PASS
Unused variables	SWC-131 SWC-135	Unused variables are allowed in Solidity and they do not pose a direct security issue.	PASS
Unexpected Ether balance	SWC-132	Contracts can behave erroneously when they strictly assume a specific Ether balance.	PASS
Hash Collisions Variable	SWC-133	Using abi.encodePacked() with multiple variable length arguments can, in certain situations, lead to a hash collision.	PASS
Hardcoded gas amount	SWC-134	The transfer() and send() functions forward a fixed amount of 2300 gas.	PASS
Unencrypted Private Data	SWC-136	It is a common misconception that private type variables cannot be read.	PASS

SMART CONTRACT ANALYSIS

Started	Sunday Sep 25 2022 08:35:59 GMT+0000 (Coordinated Universal Time)
Finished	Monday Sep 26 2022 14:31:00 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	OPINU.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-103	A FLOATING PRAGMA IS SET.	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-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 89

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
88   function add(uint256 a, uint256 b) internal pure returns (uint256) {
89   uint256 c = a + b;
90   require(c >= a, "SafeMath: addition overflow");
91   return c;
92   }
93
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 100

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
99   require(b <= a, errorMessage);
100   uint256 c = a - b;
101   return c;
102   }
103
104
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 106

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
105   if (a == 0) {return 0;}
106   uint256 c = a * b;
107   require(c / a == b, "SafeMath: multiplication overflow");
108   return c;
109   }
110
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 107

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
106   uint256 c = a * b;
107   require(c / a == b, "SafeMath: multiplication overflow");
108   return c;
109   }
110
111
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 117

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
116   require(b > 0, errorMessage);
117   uint256 c = a / b;
118   return c;
119   }
120   }
121
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 173

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
172
173   uint256 private _totalSupply = 1000000000 * (10 ** _decimals);
174   uint256 public _maxTxAmount = _totalSupply * 20 / 1000;
175   uint256 public _walletMax = _totalSupply * 20 / 1000;
176
177
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 173

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
172
173   uint256 private _totalSupply = 1000000000 * (10 ** _decimals);
174   uint256 public _maxTxAmount = _totalSupply * 20 / 1000;
175   uint256 public _walletMax = _totalSupply * 20 / 1000;
176
177
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 174

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
173   uint256 private _totalSupply = 1000000000 * (10 ** _decimals);
174   uint256 public _maxTxAmount = _totalSupply * 20 / 1000;
175   uint256 public _walletMax = _totalSupply * 20 / 1000;
176
177   address private constant DEAD_WALLET = 0x000000000000000000000000000000000000dEaD;
178
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 174

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
173   uint256 private _totalSupply = 1000000000 * (10 ** _decimals);
174   uint256 public _maxTxAmount = _totalSupply * 20 / 1000;
175   uint256 public _walletMax = _totalSupply * 20 / 1000;
176
177   address private constant DEAD_WALLET = 0x000000000000000000000000000000000000dEaD;
178
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 175

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
174   uint256 public _maxTxAmount = _totalSupply * 20 / 1000;
175   uint256 public _walletMax = _totalSupply * 20 / 1000;
176
177   address private constant DEAD_WALLET = 0x000000000000000000000000000000000000dEaD;
178   address private constant ZERO_WALLET = 0x0000000000000000000000000000000000000000;
179
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 175

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
174   uint256 public _maxTxAmount = _totalSupply * 20 / 1000;
175   uint256 public _walletMax = _totalSupply * 20 / 1000;
176
177   address private constant DEAD_WALLET = 0x000000000000000000000000000000000000dEaD;
178   address private constant ZERO_WALLET = 0x0000000000000000000000000000000000000000;
179
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 227

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
226
227   uint256 public swapThreshold = _totalSupply * 4 / 2000;
228
229   event AutoLiquify(uint256 amountBNB, uint256 amountBOG);
230
231
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 227

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
226
227   uint256 public swapThreshold = _totalSupply * 4 / 2000;
228
229   event AutoLiquify(uint256 amountBNB, uint256 amountBOG);
230
231
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 441

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
440   require(newLimit >= 5, "Wallet Limit needs to be at least 0.5%");
441   _walletMax = _totalSupply * newLimit / 1000;
442   }
443
444   function setTxLimit(uint256 newLimit) external onlyOwner {
445
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 441

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
440   require(newLimit >= 5, "Wallet Limit needs to be at least 0.5%");
441   _walletMax = _totalSupply * newLimit / 1000;
442   }
443
444   function setTxLimit(uint256 newLimit) external onlyOwner {
445
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 446

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
445   require(newLimit >= 5, "Wallet Limit needs to be at least 0.5%");
446   _maxTxAmount = _totalSupply * newLimit / 1000;
447   }
448
449   function tradingStatus(bool newStatus) public onlyOwner {
450
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 446

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
445   require(newLimit >= 5, "Wallet Limit needs to be at least 0.5%");
446   _maxTxAmount = _totalSupply * newLimit / 1000;
447   }
448
449   function tradingStatus(bool newStatus) public onlyOwner {
450
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 480

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
479   totalFee = liquidityFee.add(marketingFee).add(devFee).add(teslaFee).add(nativeFee);
480   totalFeeIfSelling = totalFee + extraSellFee;
481   require (totalFeeIfSelling + totalFee < 25);
482   }
483
484
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 481

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
480   totalFeeIfSelling = totalFee + extraSellFee;
481   require (totalFeeIfSelling + totalFee < 25);
482   }
483
484   function enable_blacklist(bool _status) public onlyOwner {
485
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 491

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
490   require(canUseBlacklist, "Can no longer add blacklists");
491   for (uint256 i; i < addresses.length; ++i) {
492   isBlacklisted[addresses[i]] = status;
493   }
494   }
495
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 544

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
543   uint256 amountETH = address(this).balance;
544   payable(msg.sender).transfer(amountETH * amountPercentage / 100);
545   }
546
547   }
548
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 544

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
543   uint256 amountETH = address(this).balance;
544   payable(msg.sender).transfer(amountETH * amountPercentage / 100);
545   }
546
547   }
548
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 5

low SEVERITY

The current pragma Solidity directive is ""^0.8.16"". 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

- OPINU.sol

Locations

```
4
5   pragma solidity ^0.8.16;
6   // SPDX-License-Identifier: Unlicensed
7
8   interface IERC20 {
9
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 397

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
396   address[] memory path = new address[](2);
397   path[0] = address(this);
398   path[1] = router.WETH();
399
400   router.swapExactTokensForETHSupportingFeeOnTransferTokens(
401
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 398

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
397   path[0] = address(this);
398   path[1] = router.WETH();
399
400   router.swapExactTokensForETHSupportingFeeOnTransferTokens(
401   amountToSwap,
402
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 492

low SEVERITY

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

Source File

- OPINU.sol

Locations

```
491   for (uint256 i; i < addresses.length; ++i) {
492   isBlacklisted[addresses[i]] = status;
493   }
494   }
495
496
```

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

LINE 304

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

- OPINU.sol

Locations

```
303   function launch() internal {
304   launchedAt = block.number;
305   }
306
307   function checkTxLimit(address sender, uint256 amount) internal view {
308
```

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.