



Love AI Token

# Smart Contract Audit Report

# TABLE OF CONTENTS

## [Audited Details](#)

- Audited Project
- Blockchain
- Addresses
- Project Website
- Codebase

## [Summary](#)

- Contract Summary
- Audit Findings Summary
- Vulnerabilities Summary

## [Conclusion](#)

## [Audit Results](#)

## [Smart Contract Analysis](#)

- Detected Vulnerabilities

## [Disclaimer](#)

## [About Us](#)

# AUDITED DETAILS

## Audited Project

Project name	Token ticker	Blockchain
Love AI Token	LoveAi	Binance Smart Chain

## Addresses

Contract address	0x3c3B0708820Ffeb88C9aFd42f0114a8480Dfe833
Contract deployer address	0xc77d62E33D5506ae61A632005D2EBcE0b95aFf26

## Project Website

<http://loveai.me/>

## Codebase

<https://bscscan.com/address/0x3c3B0708820Ffeb88C9aFd42f0114a8480Dfe833#code>

# SUMMARY

LoveAi is an Ai technology platform where everyone can send gift cards to their loved ones generated by Ai technology on our website. Everone can create a gift card for their friends,girlfriend-boyfriend, brother-sister, husband-wife, mother-father etc, through poetry and gift our \$LOVEAi token in it. We will provide the gift card on our website through Ai technology.

## Contract Summary

### **Documentation Quality**

Love AI Token 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 Love AI Token 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 445, 455, 463, 482, 484, 496, 497, 511, 513, 612, 612, 613, 694, 716, 716, 717, 732 and 752.
- 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 742 and 743.

## CONCLUSION

We have audited the Love AI Token project released on February 2023 to discover issues and identify potential security vulnerabilities in Love AI Token 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 Love AI Token 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, a state variable visibility is not 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.	PASS
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.	PASS
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 <code>abi.encodePacked()</code> with multiple variable length arguments can, in certain situations, lead to a hash collision.	PASS
Hardcoded gas amount	SWC-134	The <code>transfer()</code> and <code>send()</code> 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	Saturday Feb 04 2023 12:45:03 GMT+0000 (Coordinated Universal Time)
Finished	Sunday Feb 05 2023 01:12:54 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	LoveAi.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-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 445

## low SEVERITY

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

## Source File

- LoveAi.sol

## Locations

```
444     unchecked {  
445         _approve(sender, _msgSender(), currentAllowance - amount);  
446     }  
447 }  
448  
449
```

## SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 455

### low SEVERITY

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

### Source File

- LoveAi.sol

### Locations

```
454 function increaseAllowance(address spender, uint256 addedValue) public virtual
returns (bool) {
455     _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
456     return true;
457 }
458
459
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 463

## low SEVERITY

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

## Source File

- LoveAi.sol

## Locations

```
462     unchecked {
463         _approve(_msgSender(), spender, currentAllowance - subtractedValue);
464     }
465
466     return true;
467
```

## SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 482

### low SEVERITY

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

### Source File

- LoveAi.sol

### Locations

```
481     unchecked {  
482         _balances[sender] = senderBalance - amount;  
483     }  
484     _balances[recipient] += amount;  
485  
486
```

## SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 484

### low SEVERITY

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

### Source File

- LoveAi.sol

### Locations

```
483     }  
484     _balances[recipient] += amount;  
485  
486     emit Transfer(sender, recipient, amount);  
487  
488
```

## SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 496

### low SEVERITY

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

### Source File

- LoveAi.sol

### Locations

```
495
496   _totalSupply += amount;
497   _balances[account] += amount;
498   emit Transfer(address(0), account, amount);
499
500
```



# SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 497

## low SEVERITY

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

## Source File

- LoveAi.sol

## Locations

```
496     _totalSupply += amount;
497     _balances[account] += amount;
498     emit Transfer(address(0), account, amount);
499
500     _afterTokenTransfer(address(0), account, amount);
501
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 511

## low SEVERITY

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

## Source File

- LoveAi.sol

## Locations

```
510     unchecked {  
511         _balances[account] = accountBalance - amount;  
512     }  
513     _totalSupply -= amount;  
514  
515
```

## SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 513

### low SEVERITY

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

### Source File

- LoveAi.sol

### Locations

```
512     }  
513     _totalSupply -= amount;  
514  
515     emit Transfer(account, address(0), amount);  
516  
517
```

# SWC-101 | ARITHMETIC OPERATION "\*" DISCOVERED

LINE 612

## low SEVERITY

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

## Source File

- LoveAi.sol

## Locations

```
611
612  _mint(owner(), 5e8 * (10 ** decimals()));
613  swapTokensAtAmount = totalSupply() / 5_000;
614
615  swapEnabled = true;
616
```

## SWC-101 | ARITHMETIC OPERATION "\*\*" DISCOVERED

LINE 612

### low SEVERITY

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

### Source File

- LoveAi.sol

### Locations

```
611
612  _mint(owner(), 5e8 * (10 ** decimals()));
613  swapTokensAtAmount = totalSupply() / 5_000;
614
615  swapEnabled = true;
616
```

## SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 613

### low SEVERITY

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

### Source File

- LoveAi.sol

### Locations

```
612  _mint(owner(), 5e8 * (10 ** decimals()));
613  swapTokensAtAmount = totalSupply() / 5_000;
614
615  swapEnabled = true;
616  }
617
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 694

## low SEVERITY

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

## Source File

- LoveAi.sol

## Locations

```
693     to == uniswapV2Pair &&  
694     feeOnBuy + feeOnSell > 0 &&  
695     swapEnabled  
696   ) {  
697     swapping = true;  
698
```

## SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 716

### low SEVERITY

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

### Source File

- LoveAi.sol

### Locations

```
715     if (_totalFees > 0) {  
716         uint256 fees = (amount * _totalFees) / 100;  
717         amount = amount - fees;  
718         super._transfer(from, address(this), fees);  
719     }  
720
```



## SWC-101 | ARITHMETIC OPERATION "\*" DISCOVERED

LINE 716

### low SEVERITY

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

### Source File

- LoveAi.sol

### Locations

```
715     if (_totalFees > 0) {  
716         uint256 fees = (amount * _totalFees) / 100;  
717         amount = amount - fees;  
718         super._transfer(from, address(this), fees);  
719     }  
720
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 717

## low SEVERITY

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

## Source File

- LoveAi.sol

## Locations

```
716 uint256 fees = (amount * _totalFees) / 100;
717 amount = amount - fees;
718 super._transfer(from, address(this), fees);
719 }
720
721
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 732

## low SEVERITY

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

## Source File

- LoveAi.sol

## Locations

```
731 function setSwapTokensAtAmount(uint256 newAmount) external onlyOwner{
732     require(newAmount > totalSupply() / 1_000_000, "SwapTokensAtAmount must be greater
than 0.0001% of total supply");
733     swapTokensAtAmount = newAmount;
734
735     emit SwapTokensAtAmountUpdated(swapTokensAtAmount);
736
```

## SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 752

### low SEVERITY

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

### Source File

- LoveAi.sol

### Locations

```
751
752     uint256 newBalance = address(this).balance - initialBalance;
753     payable(marketingWallet).sendValue(newBalance);
754     emit SwapAndSendFee(tokenAmount, newBalance);
755 }
756
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 742

### low SEVERITY

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

### Source File

- LoveAi.sol

### Locations

```
741 address[] memory path = new address[](2);
742 path[0] = address(this);
743 path[1] = uniswapV2Router.WETH();
744
745 uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
746
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 743

### low SEVERITY

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

### Source File

- LoveAi.sol

### Locations

```
742 path[0] = address(this);
743 path[1] = uniswapV2Router.WETH();
744
745 uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
746 tokenAmount,
747
```

# 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.