



WeSendit

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
WeSendit	WSI	Binance Smart Chain

Addresses

Contract address	0x837a130aed114300bab4f9f1f4f500682f7efd48
Contract deployer address	0x7D48d8F61b1038C2B53D5d157766C92e69ba2Ea7

Project Website

<https://wesendit.io/>

Codebase

<https://bscscan.com/address/0x837a130aed114300bab4f9f1f4f500682f7efd48#code>

SUMMARY

Since the launch of WeSendit.com in 2013, we have built a professional and efficient structure and analyzed – and evaluated – comprehensive fundamental data from centralized and decentralized application areas. We will develop our unique selling propositions and market them globally.

Contract Summary

Documentation Quality

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

- The technical description is provided clearly and structured and also don't have any high risk issue.

Code Quality

The Overall quality of the basecode is standard.

- Standard solidity basecode and rules are already followed by WeSendit 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 1062, 1063, 1067, 1068, 1068, 1069, 1084, 1094, 1094, 1097, 1097, 1097, 1454, 1455, 2223, 2224, 2236, 2247, 2310, 2377, 2377, 2654, 2668, 2706, 2711, 2714, 2721, 2752, 2794, 2882, 2882, 2939, 3160, 3183, 3216, 3218, 3239, 3240, 3265, 3267, 3316, 3396, 1454, 1455, 2654 and 2668.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 56, 760, 827, 919, 952, 980, 1009, 1040, 1119, 1368, 1739, 1805, 2953, 2983, 3368 and 3407.
- 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 1068, 1095, 1096, 1098, 1098, 1458, 1461, 1503, 2006, 2259, 2260, 2289, 2290, 2664, 2668, 2668, 2707, 2940, 2940, 3691, 3700 and 3710.

CONCLUSION

We have audited the WeSendit project released on October 2022 to discover issues and identify potential security vulnerabilities in WeSendit 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 WeSendit 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, and out-of-bounds array access which the index access expression can cause an exception in case an invalid array index value is used. The current pragma Solidity directive is `""^0.8.0""`. Specifying a fixed compiler version is recommended 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.

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.	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 grieving 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	Friday Oct 28 2022 12:50:09 GMT+0000 (Coordinated Universal Time)
Finished	Saturday Oct 29 2022 22:54:19 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	WeSenditToken.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	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-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" 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-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1062

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1061 while (temp != 0) {  
1062     digits++;  
1063     temp /= 10;  
1064 }  
1065 bytes memory buffer = new bytes(digits);  
1066
```

SWC-101 | ARITHMETIC OPERATION "/=" DISCOVERED

LINE 1063

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1062  digits++;  
1063  temp /= 10;  
1064  }  
1065  bytes memory buffer = new bytes(digits);  
1066  while (value != 0) {  
1067
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 1067

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1066 while (value != 0) {  
1067     digits -= 1;  
1068     buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));  
1069     value /= 10;  
1070 }  
1071
```


SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1068

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1067     digits -= 1;
1068     buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
1069     value /= 10;
1070 }
1071 return string(buffer);
1072
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 1068

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1067     digits -= 1;
1068     buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
1069     value /= 10;
1070 }
1071 return string(buffer);
1072
```

SWC-101 | ARITHMETIC OPERATION "/=" DISCOVERED

LINE 1069

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1068     buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
1069     value /= 10;
1070 }
1071 return string(buffer);
1072 }
1073
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1084

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1083 while (temp != 0) {  
1084     length++;  
1085     temp >>= 8;  
1086 }  
1087 return toHexString(value, length);  
1088
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1094

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1093 function toHexString(uint256 value, uint256 length) internal pure returns (string
memory) {
1094     bytes memory buffer = new bytes(2 * length + 2);
1095     buffer[0] = "0";
1096     buffer[1] = "x";
1097     for (uint256 i = 2 * length + 1; i > 1; --i) {
1098
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1094

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1093 function toHexString(uint256 value, uint256 length) internal pure returns (string
memory) {
1094     bytes memory buffer = new bytes(2 * length + 2);
1095     buffer[0] = "0";
1096     buffer[1] = "x";
1097     for (uint256 i = 2 * length + 1; i > 1; --i) {
1098
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1097

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1096     buffer[1] = "x";
1097     for (uint256 i = 2 * length + 1; i > 1; --i) {
1098         buffer[i] = _HEX_SYMBOLS[value & 0xf];
1099         value >>= 4;
1100     }
1101
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1097

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1096     buffer[1] = "x";
1097     for (uint256 i = 2 * length + 1; i > 1; --i) {
1098         buffer[i] = _HEX_SYMBOLS[value & 0xf];
1099         value >>= 4;
1100     }
1101
```


SWC-101 | ARITHMETIC OPERATION "--" DISCOVERED

LINE 1097

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1096     buffer[1] = "x";
1097     for (uint256 i = 2 * length + 1; i > 1; --i) {
1098         buffer[i] = _HEX_SYMBOLS[value & 0xf];
1099         value >>= 4;
1100     }
1101
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1454

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1453
1454     uint256 toDeleteIndex = valueIndex - 1;
1455     uint256 lastIndex = set._values.length - 1;
1456
1457     if (lastIndex != toDeleteIndex) {
1458
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1455

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1454     uint256 toDeleteIndex = valueIndex - 1;
1455     uint256 lastIndex = set._values.length - 1;
1456
1457     if (lastIndex != toDeleteIndex) {
1458         bytes32 lastValue = set._values[lastIndex];
1459     }
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 2223

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2222 // split the contract balance into halves
2223 uint256 half = amount / 2;
2224 uint256 otherHalf = amount - half;
2225
2226 // capture the contract's current BNB balance.
2227
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2224

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2223     uint256 half = amount / 2;
2224     uint256 otherHalf = amount - half;
2225
2226     // capture the contract's current BNB balance.
2227     // this is so that we can capture exactly the amount of BNB that the
2228
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2236

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2235 // how much BNB did we just swap into?
2236 uint256 newBalance = address(this).balance - initialBalance;
2237
2238 // add liquidity to uniswap
2239 uint256 tokenLiquified = _addLiquidity(
2240
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 2247

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2246
2247     return half + tokenLiquified;
2248 }
2249
2250 /**
2251
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2310

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2309 // how much BUSD did we just swap into?
2310 uint256 newBalance = IERC20(busdAddress()).balanceOf(destination) -
2311 initialBalance;
2312
2313 emit SwapTokenForBusd(
2314
```


SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 2377

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2376 // Calculate percentual amount of volume
2377 uint256 percentualAmount = (pancakePairTokenBalance *
2378 percentageVolume) / 100;
2379
2380 // Do not exceed swap or liquify amount from fee entry
2381
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 2377

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2376 // Calculate percentual amount of volume
2377 uint256 percentualAmount = (pancakePairTokenBalance *
2378 percentageVolume) / 100;
2379
2380 // Do not exceed swap or liquify amount from fee entry
2381
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2654

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2653 // Return entry index
2654 return feeEntries.length - 1;
2655 }
2656
2657 function removeFee(uint256 index) external override onlyRole(ADMIN) {
2658
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2668

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2667 // Remove fee entry from array
2668 feeEntries[index] = feeEntries[feeEntries.length - 1];
2669 feeEntries.pop();
2670
2671 emit FeeRemoved(id, index);
2672
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 2706

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2705
2706   for (uint256 i = 0; i < feeAmount; i++) {
2707       FeeEntry memory fee = feeEntries[i];
2708
2709       if (_isFeeEntryValid(fee) && _isFeeEntryMatching(fee, from, to)) {
2710
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 2711

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2710  uint256 tFee = _calculateFee(amount, fee.percentage);
2711  uint256 tempPercentage = totalFeePercentage + fee.percentage;
2712
2713  if (tFee > 0 && tempPercentage <= txFeeLimit) {
2714    tFees = tFees + tFee;
2715
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 2714

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2713     if (tFee > 0 && tempPercentage <= txFeeLimit) {  
2714         tFees = tFees + tFee;  
2715         totalFeePercentage = tempPercentage;  
2716         _reflectFee(from, to, tFee, fee, bypassSwapAndLiquify);  
2717     }  
2718
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2721

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2720
2721     tTotal = amount - tFees;
2722     require(tTotal > 0, "DynamicFeeManager: invalid total amount");
2723
2724     return (tTotal, tFees);
2725
```


SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 2752

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2751     );  
2752     feeEntryAmounts[fee.id] = feeEntryAmounts[fee.id] + tFee;  
2753 } else {  
2754     require(  
2755         IWeSenditToken(address(token())).transferFromNoFees(  
2756
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2794

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2793 // Subtract amount of swapped token from fee entry amount
2794 feeEntryAmounts[fee.id] = feeEntryAmounts[fee.id] - tokenSwapped;
2795 }
2796
2797 // Check if callback should be called on destination
2798
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 2882

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2881 {  
2882     return (amount * percentage) / FEE_DIVIDER;  
2883 }  
2884  
2885 /**  
2886
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 2882

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2881 {  
2882   return (amount * percentage) / FEE_DIVIDER;  
2883 }  
2884  
2885 /**  
2886
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 2939

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2938
2939   for (uint256 i = 0; i < addresses.length; i++) {
2940       require(_token.transfer(addresses[i], amounts[i]));
2941   }
2942
2943
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 3160

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
3159     address owner = _msgSender();
3160     _approve(owner, spender, allowance(owner, spender) + addedValue);
3161     return true;
3162 }
3163
3164
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 3183

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
3182     unchecked {  
3183         _approve(owner, spender, currentAllowance - subtractedValue);  
3184     }  
3185  
3186     return true;  
3187
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 3216

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
3215 unchecked {  
3216   _balances[from] = fromBalance - amount;  
3217 }  
3218 _balances[to] += amount;  
3219  
3220
```


SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 3218

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
3217     }  
3218     _balances[to] += amount;  
3219  
3220     emit Transfer(from, to, amount);  
3221  
3222
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 3239

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
3238
3239 _totalSupply += amount;
3240 _balances[account] += amount;
3241 emit Transfer(address(0), account, amount);
3242
3243
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 3240

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
3239 _totalSupply += amount;  
3240 _balances[account] += amount;  
3241 emit Transfer(address(0), account, amount);  
3242  
3243 _afterTokenTransfer(address(0), account, amount);  
3244
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 3265

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
3264 unchecked {  
3265   _balances[account] = accountBalance - amount;  
3266 }  
3267 _totalSupply -= amount;  
3268  
3269
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 3267

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
3266     }  
3267     _totalSupply -= amount;  
3268  
3269     emit Transfer(account, address(0), amount);  
3270  
3271
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 3316

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
3315     unchecked {  
3316         _approve(owner, spender, currentAllowance - amount);  
3317     }  
3318 }  
3319 }  
3320
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 3396

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
3395 function _mint(address account, uint256 amount) internal virtual override {  
3396     require(ERC20.totalSupply() + amount <= cap(), "ERC20Capped: cap exceeded");  
3397     super._mint(account, amount);  
3398 }  
3399 }  
3400
```

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

LINE 1454

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1453
1454     uint256 toDeleteIndex = valueIndex - 1;
1455     uint256 lastIndex = set._values.length - 1;
1456
1457     if (lastIndex != toDeleteIndex) {
1458
```


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

LINE 1455

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1454     uint256 toDeleteIndex = valueIndex - 1;
1455     uint256 lastIndex = set._values.length - 1;
1456
1457     if (lastIndex != toDeleteIndex) {
1458         bytes32 lastValue = set._values[lastIndex];
1459     }
```

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

LINE 2654

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2653 // Return entry index
2654 return feeEntries.length - 1;
2655 }
2656
2657 function removeFee(uint256 index) external override onlyRole(ADMIN) {
2658
```

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

LINE 2668

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2667 // Remove fee entry from array
2668 feeEntries[index] = feeEntries[feeEntries.length - 1];
2669 feeEntries.pop();
2670
2671 emit FeeRemoved(id, index);
2672
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 56

low SEVERITY

The current pragma Solidity directive is `""^0.8.0""`. 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

- WeSenditToken.sol

Locations

```
55
56  pragma solidity ^0.8.0;
57
58  /**
59   * @dev Interface of the ERC20 standard as defined in the EIP.
60
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 760

low SEVERITY

The current pragma Solidity directive is `""^0.8.0""`. 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

- WeSenditToken.sol

Locations

```
759
760 pragma solidity ^0.8.0;
761
762 /**
763  * @dev Contract module that helps prevent reentrant calls to a function.
764
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 827

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". 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

- WeSenditToken.sol

Locations

```
826
827  pragma solidity ^0.8.0;
828
829  /**
830   * @dev External interface of AccessControl declared to support ERC165 detection.
831
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 919

low SEVERITY

The current pragma Solidity directive is `""^0.8.0""`. 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

- WeSenditToken.sol

Locations

```
918
919  pragma solidity ^0.8.0;
920
921  /**
922   * @dev External interface of AccessControlEnumerable declared to support ERC165
923   detection.
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 952

low SEVERITY

The current pragma Solidity directive is `""^0.8.0""`. 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

- WeSenditToken.sol

Locations

```
951
952  pragma solidity ^0.8.0;
953
954  /**
955   * @dev Provides information about the current execution context, including the
956
```


SWC-103 | A FLOATING PRAGMA IS SET.

LINE 980

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". 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

- WeSenditToken.sol

Locations

```
979
980 pragma solidity ^0.8.0;
981
982 /**
983  * @dev Interface of the ERC165 standard, as defined in the
984
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1009

low SEVERITY

The current pragma Solidity directive is `""^0.8.0""`. 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

- WeSenditToken.sol

Locations

```
1008
1009  pragma solidity ^0.8.0;
1010
1011  /**
1012   * @dev Implementation of the {IERC165} interface.
1013
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1040

low SEVERITY

The current pragma Solidity directive is `""^0.8.0""`. 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

- WeSenditToken.sol

Locations

```
1039
1040  pragma solidity ^0.8.0;
1041
1042  /**
1043   * @dev String operations.
1044
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1119

low SEVERITY

The current pragma Solidity directive is `""^0.8.0""`. 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

- WeSenditToken.sol

Locations

```
1118
1119  pragma solidity ^0.8.0;
1120
1121
1122
1123
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1368

low SEVERITY

The current pragma Solidity directive is `""^0.8.0""`. 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

- WeSenditToken.sol

Locations

```
1367
1368  pragma solidity ^0.8.0;
1369
1370  /**
1371   * @dev Library for managing
1372
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1739

low SEVERITY

The current pragma Solidity directive is `""^0.8.0""`. 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

- WeSenditToken.sol

Locations

```
1738
1739  pragma solidity ^0.8.0;
1740
1741
1742
1743
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1805

low SEVERITY

The current pragma Solidity directive is `""^0.8.0""`. 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

- WeSenditToken.sol

Locations

```
1804
1805  pragma solidity ^0.8.0;
1806
1807  /**
1808   * @dev Contract module which provides a basic access control mechanism, where
1809
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 2953

low SEVERITY

The current pragma Solidity directive is `""^0.8.0""`. 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

- WeSenditToken.sol

Locations

```
2952
2953  pragma solidity ^0.8.0;
2954
2955  /**
2956   * @dev Interface for the optional metadata functions from the ERC20 standard.
2957
```


SWC-103 | A FLOATING PRAGMA IS SET.

LINE 2983

low SEVERITY

The current pragma Solidity directive is `""^0.8.0""`. 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

- WeSenditToken.sol

Locations

```
2982
2983  pragma solidity ^0.8.0;
2984
2985
2986
2987
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 3368

low SEVERITY

The current pragma Solidity directive is `""^0.8.0""`. 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

- WeSenditToken.sol

Locations

```
3367
3368  pragma solidity ^0.8.0;
3369
3370  /**
3371   * @dev Extension of {ERC20} that adds a cap to the supply of tokens.
3372
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 3407

low SEVERITY

The current pragma Solidity directive is `""^0.8.0""`. 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

- WeSenditToken.sol

Locations

```
3406
3407  pragma solidity ^0.8.0;
3408
3409
3410  /**
3411
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1068

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1067     digits -= 1;
1068     buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
1069     value /= 10;
1070 }
1071 return string(buffer);
1072
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1095

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1094 bytes memory buffer = new bytes(2 * length + 2);
1095 buffer[0] = "0";
1096 buffer[1] = "x";
1097 for (uint256 i = 2 * length + 1; i > 1; --i) {
1098     buffer[i] = _HEX_SYMBOLS[value & 0xf];
1099 }
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1096

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1095     buffer[0] = "0";
1096     buffer[1] = "x";
1097     for (uint256 i = 2 * length + 1; i > 1; --i) {
1098         buffer[i] = _HEX_SYMBOLS[value & 0xf];
1099         value >>= 4;
1100     }
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1098

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1097   for (uint256 i = 2 * length + 1; i > 1; --i) {  
1098     buffer[i] = _HEX_SYMBOLS[value & 0xf];  
1099     value >>= 4;  
1100   }  
1101   require(value == 0, "Strings: hex length insufficient");  
1102
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1098

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1097   for (uint256 i = 2 * length + 1; i > 1; --i) {  
1098     buffer[i] = _HEX_SYMBOLS[value & 0xf];  
1099     value >>= 4;  
1100   }  
1101   require(value == 0, "Strings: hex length insufficient");  
1102
```


SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1458

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1457   if (lastIndex != toDeleteIndex) {  
1458       bytes32 lastValue = set._values[lastIndex];  
1459  
1460       // Move the last value to the index where the value to delete is  
1461       set._values[toDeleteIndex] = lastValue;  
1462   }
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1461

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1460 // Move the last value to the index where the value to delete is
1461 set._values[toDeleteIndex] = lastValue;
1462 // Update the index for the moved value
1463 set._indexes[lastValue] = valueIndex; // Replace lastValue's index to valueIndex
1464 }
1465
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1503

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
1502     function _at(Set storage set, uint256 index) private view returns (bytes32) {  
1503         return set._values[index];  
1504     }  
1505  
1506     /**  
1507
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2006

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2005  {  
2006    return feeEntries[index];  
2007  }  
2008  
2009  function getFeeAmount(bytes32 id)  
2010
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2259

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2258 address[] memory path = new address[](2);
2259 path[0] = address(token());
2260 path[1] = pancakeRouter().WETH();
2261
2262 require(
2263
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2260

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2259 path[0] = address(token());  
2260 path[1] = pancakeRouter().WETH();  
2261  
2262 require(  
2263 token().approve(address(pancakeRouter()), amount),  
2264
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2289

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2288     address[] memory path = new address[](2);
2289     path[0] = address(token());
2290     path[1] = busdAddress();
2291
2292     require(
2293
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2290

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2289 path[0] = address(token());  
2290 path[1] = busdAddress();  
2291  
2292 require(  
2293 token().approve(address(pancakeRouter()), amount),  
2294
```


SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2664

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2663 // Reset current amount for liquify or swap
2664 bytes32 id = feeEntries[index].id;
2665 feeEntryAmounts[id] = 0;
2666
2667 // Remove fee entry from array
2668
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2668

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2667 // Remove fee entry from array
2668 feeEntries[index] = feeEntries[feeEntries.length - 1];
2669 feeEntries.pop();
2670
2671 emit FeeRemoved(id, index);
2672
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2668

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2667 // Remove fee entry from array
2668 feeEntries[index] = feeEntries[feeEntries.length - 1];
2669 feeEntries.pop();
2670
2671 emit FeeRemoved(id, index);
2672
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2707

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2706   for (uint256 i = 0; i < feeAmount; i++) {  
2707       FeeEntry memory fee = feeEntries[i];  
2708  
2709       if (_isFeeEntryValid(fee) && _isFeeEntryMatching(fee, from, to)) {  
2710           uint256 tFee = _calculateFee(amount, fee.percentage);  
2711
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2940

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2939   for (uint256 i = 0; i < addresses.length; i++) {  
2940       require(_token.transfer(addresses[i], amounts[i]));  
2941   }  
2942  
2943   return true;  
2944
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2940

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
2939   for (uint256 i = 0; i < addresses.length; i++) {  
2940       require(_token.transfer(addresses[i], amounts[i]));  
2941   }  
2942  
2943   return true;  
2944
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 3691

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
3690     ) public {  
3691         IERC20(path[0]).transferFrom(msg.sender, _pair, amountIn);  
3692     }  
3693  
3694     function swapExactETHForTokensSupportingFeeOnTransferTokens(  
3695
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 3700

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
3699     ) payable {  
3700     MockPancakePair(_pair).swap(path[1], msg.sender, amountOutMin);  
3701     }  
3702  
3703     function swapExactTokensForTokensSupportingFeeOnTransferTokens(  
3704
```


SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 3710

low SEVERITY

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

Source File

- WeSenditToken.sol

Locations

```
3709     ) public {  
3710         IERC20(path[0]).transferFrom(msg.sender, _pair, amountIn);  
3711     }  
3712 }  
3713
```

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.