Shinjiru Inu

# Smart Contract Audit Report

SYSFIXED

21 Jan 2003

# TABLE OF CONTENTS

# AUDITED DETAILS

## Audited Project

| Project name | Token ticker | Blockchain |
|---|---|---|
| Shinjiru Inu | SHINJI | Binance Smart Chain |

## Addresses

| | |
|---|---|
| **Contract address** | 0x87e0ce18ce0ce0a86b22537b48c15e03a519b112 |
| **Contract deployer address** | 0xe3930701d4d74a5CfD6438A37ACd25b078D358bd |

## Project Website

**https://github.com/ShinjiruInu**

## Codebase

**https://bscscan.com/address/0x87e0ce18ce0ce0a86b22537b48c15e03a519b112#contracts**

# SUMMARY

The Shinjiru Swap is a secure and reliable platform for investors to easily swap their tokens. It is powered by a peer-to-peer protocol, allowing users to securely and anonymously exchange tokens without having to trust a third party. Additionally, the swap is integrated with the Shinjiru Multi-Chain Staking Pools, allowing investors to quickly and easily stake their tokens and earn rewards. It is easy to use and accessible to all investors, regardless of their experience. The swap is also highly liquid, allowing investors to quickly and easily convert their holdings into other digital assets. Furthermore, the platform has low fees, allowing investors to maximize their profits. Finally, the Shinjiru Swap is backed by the Shinjiru team, providing investors with peace of mind that their funds are safe and secure. Using the Shinjiru Swap is relatively straightforward. First, investors will need to connect their wallets that hold their Shinjiru Tokens to the Shinjiru Swap. After that, they can select the tokens they wish to exchange and set their own exchange rate. Once the transaction is complete, the tokens will be credited to the investor's wallet. Finally, the investor can use the tokens to stake and earn rewards.

## Contract Summary

**Documentation Quality**

Shinjiru 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 Shinjiru 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 25, 29, 33, 37, 43, 50, 363, 363, 363, 378, 378, 381, 381, 505, 538, 554, 554, 555, 556, 557, 559, 559, 560, 560, 561, 561, 564, 564, 565, 567, 567, 567, 567, 568, 568, 570, 570, 570, 570, 571, 571, 574, 574, 575, 575, 616, 616, 627, 628, 632, 636, 636, 637, 639, 640, 641, 645, 649, 649, 650, 652, 653, 654 and 658.
- 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 587 and 588.

# CONCLUSION

We have audited the Shinjiru Inu project released on January 2023 to discover issues and identify potential security vulnerabilities in Shinjiru 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 Shinjiru 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 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<br>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<br>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<br>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<br>SWC-121<br>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<br>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** | Tuesday Jan 21 2003 11:48:55 GMT+0000 (Coordinated Universal Time) |
| **Finished** | Wednesday Jan 22 2003 10:54:22 GMT+0000 (Coordinated Universal Time) |
| **Mode** | Standard |
| **Main Source File** | Shinjiru.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 | 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-110 | OUT OF BOUNDS ARRAY ACCESS | low | acknowledged |
| SWC-110 | OUT OF BOUNDS ARRAY ACCESS | low | acknowledged |

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 25

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
24   function add(uint256 a, uint256 b) internal pure returns (uint256) {
25   return a + b;
26   }
27
28   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
29
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 29

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
28   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
29   return a - b;
30   }
31
32   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
33
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 33

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
32   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
33   return a * b;
34   }
35
36   function div(uint256 a, uint256 b) internal pure returns (uint256) {
37
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 37

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
36   function div(uint256 a, uint256 b) internal pure returns (uint256) {
37   return a / b;
38   }
39
40   function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns
(uint256) {
41
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 43

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
42   require(b <= a, errorMessage);
43   return a - b;
44   }
45   }
46
47
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 50

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
49   require(b > 0, errorMessage);
50   return a / b;
51   }
52   }
53
54
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 363

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
362   uint8 private constant _decimals = 9;
363   uint256 private _tTotal = 10**15 * 10**_decimals;
364   string private constant _name = "Shinjiru Inu";
365   string private constant _symbol = unicode"SHINJI";
366
367
```

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

LINE 363

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
362   uint8 private constant _decimals = 9;
363   uint256 private _tTotal = 10**15 * 10**_decimals;
364   string private constant _name = "Shinjiru Inu";
365   string private constant _symbol = unicode"SHINJI";
366
367
```

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

LINE 363

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
362   uint8 private constant _decimals = 9;
363   uint256 private _tTotal = 10**15 * 10**_decimals;
364   string private constant _name = "Shinjiru Inu";
365   string private constant _symbol = unicode"SHINJI";
366
367
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 378

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
377
378   uint256 public _maxWalletToken = _tTotal * 100 / 100;
379   uint256 private _previousMaxWalletToken = _maxWalletToken;
380
381   uint256 public _maxTxAmount = _tTotal * 100 / 100;
382
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 378

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
377
378   uint256 public _maxWalletToken = _tTotal * 100 / 100;
379   uint256 private _previousMaxWalletToken = _maxWalletToken;
380
381   uint256 public _maxTxAmount = _tTotal * 100 / 100;
382
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 381

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
380
381   uint256 public _maxTxAmount = _tTotal * 100 / 100;
382   uint256 private _previousMaxTxAmount = _maxTxAmount;
383
384
385
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 381

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
380
381   uint256 public _maxTxAmount = _tTotal * 100 / 100;
382   uint256 private _previousMaxTxAmount = _maxTxAmount;
383
384
385
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 505

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
504   uint256 heldTokens = balanceOf(to);
505   require((heldTokens + amount) <= _maxWalletToken,"Over wallet limit.");}
506
507   if (from != owner())
508   require(amount <= _maxTxAmount, "Over transaction limit.");
509
```

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

LINE 538

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
537
538   txCount++;
539
540   }
541
542
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 554

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
553
554   uint256 tokens_to_Burn = contractTokenBalance * Percent_Burn / 100;
555   _tTotal = _tTotal - tokens_to_Burn;
556   _tOwned[Wallet_Burn] = _tOwned[Wallet_Burn] + tokens_to_Burn;
557   _tOwned[address(this)] = _tOwned[address(this)] - tokens_to_Burn;
558
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 554

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
553
554   uint256 tokens_to_Burn = contractTokenBalance * Percent_Burn / 100;
555   _tTotal = _tTotal - tokens_to_Burn;
556   _tOwned[Wallet_Burn] = _tOwned[Wallet_Burn] + tokens_to_Burn;
557   _tOwned[address(this)] = _tOwned[address(this)] - tokens_to_Burn;
558
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 555

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
554   uint256 tokens_to_Burn = contractTokenBalance * Percent_Burn / 100;
555   _tTotal = _tTotal - tokens_to_Burn;
556   _tOwned[Wallet_Burn] = _tOwned[Wallet_Burn] + tokens_to_Burn;
557   _tOwned[address(this)] = _tOwned[address(this)] - tokens_to_Burn;
558
559
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 556

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
555   _tTotal = _tTotal - tokens_to_Burn;
556   _tOwned[Wallet_Burn] = _tOwned[Wallet_Burn] + tokens_to_Burn;
557   _tOwned[address(this)] = _tOwned[address(this)] - tokens_to_Burn;
558
559   uint256 tokens_to_M = contractTokenBalance * Percent_Marketing / 100;
560
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 557

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
556   _tOwned[Wallet_Burn] = _tOwned[Wallet_Burn] + tokens_to_Burn;
557   _tOwned[address(this)] = _tOwned[address(this)] - tokens_to_Burn;
558
559   uint256 tokens_to_M = contractTokenBalance * Percent_Marketing / 100;
560   uint256 tokens_to_D = contractTokenBalance * Percent_Dev / 100;
561
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 559

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
558
559   uint256 tokens_to_M = contractTokenBalance * Percent_Marketing / 100;
560   uint256 tokens_to_D = contractTokenBalance * Percent_Dev / 100;
561   uint256 tokens_to_LP_Half = contractTokenBalance * Percent_AutoLP / 200;
562
563
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 559

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
558
559   uint256 tokens_to_M = contractTokenBalance * Percent_Marketing / 100;
560   uint256 tokens_to_D = contractTokenBalance * Percent_Dev / 100;
561   uint256 tokens_to_LP_Half = contractTokenBalance * Percent_AutoLP / 200;
562
563
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 560

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
559   uint256 tokens_to_M = contractTokenBalance * Percent_Marketing / 100;
560   uint256 tokens_to_D = contractTokenBalance * Percent_Dev / 100;
561   uint256 tokens_to_LP_Half = contractTokenBalance * Percent_AutoLP / 200;
562
563   uint256 balanceBeforeSwap = address(this).balance;
564
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 560

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
559   uint256 tokens_to_M = contractTokenBalance * Percent_Marketing / 100;
560   uint256 tokens_to_D = contractTokenBalance * Percent_Dev / 100;
561   uint256 tokens_to_LP_Half = contractTokenBalance * Percent_AutoLP / 200;
562
563   uint256 balanceBeforeSwap = address(this).balance;
564
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 561

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
560   uint256 tokens_to_D = contractTokenBalance * Percent_Dev / 100;
561   uint256 tokens_to_LP_Half = contractTokenBalance * Percent_AutoLP / 200;
562
563   uint256 balanceBeforeSwap = address(this).balance;
564   swapTokensForBNB(tokens_to_LP_Half + tokens_to_M + tokens_to_D);
565
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 561

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
560   uint256 tokens_to_D = contractTokenBalance * Percent_Dev / 100;
561   uint256 tokens_to_LP_Half = contractTokenBalance * Percent_AutoLP / 200;
562
563   uint256 balanceBeforeSwap = address(this).balance;
564   swapTokensForBNB(tokens_to_LP_Half + tokens_to_M + tokens_to_D);
565
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 564

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
563   uint256 balanceBeforeSwap = address(this).balance;
564   swapTokensForBNB(tokens_to_LP_Half + tokens_to_M + tokens_to_D);
565   uint256 BNB_Total = address(this).balance - balanceBeforeSwap;
566
567   uint256 split_M = Percent_Marketing * 100 / (Percent_AutoLP + Percent_Marketing +
Percent_Dev);
568
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 564

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
563   uint256 balanceBeforeSwap = address(this).balance;
564   swapTokensForBNB(tokens_to_LP_Half + tokens_to_M + tokens_to_D);
565   uint256 BNB_Total = address(this).balance - balanceBeforeSwap;
566
567   uint256 split_M = Percent_Marketing * 100 / (Percent_AutoLP + Percent_Marketing +
Percent_Dev);
568
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 565

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
564   swapTokensForBNB(tokens_to_LP_Half + tokens_to_M + tokens_to_D);
565   uint256 BNB_Total = address(this).balance - balanceBeforeSwap;
566
567   uint256 split_M = Percent_Marketing * 100 / (Percent_AutoLP + Percent_Marketing +
Percent_Dev);
568   uint256 BNB_M = BNB_Total * split_M / 100;
569
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 567

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
566
567   uint256 split_M = Percent_Marketing * 100 / (Percent_AutoLP + Percent_Marketing +
Percent_Dev);
568   uint256 BNB_M = BNB_Total * split_M / 100;
569
570   uint256 split_D = Percent_Dev * 100 / (Percent_AutoLP + Percent_Marketing +
Percent_Dev);
571
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 567

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
566
567   uint256 split_M = Percent_Marketing * 100 / (Percent_AutoLP + Percent_Marketing +
Percent_Dev);
568   uint256 BNB_M = BNB_Total * split_M / 100;
569
570   uint256 split_D = Percent_Dev * 100 / (Percent_AutoLP + Percent_Marketing +
Percent_Dev);
571
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 567

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
566
567   uint256 split_M = Percent_Marketing * 100 / (Percent_AutoLP + Percent_Marketing +
Percent_Dev);
568   uint256 BNB_M = BNB_Total * split_M / 100;
569
570   uint256 split_D = Percent_Dev * 100 / (Percent_AutoLP + Percent_Marketing +
Percent_Dev);
571
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 567

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
566
567   uint256 split_M = Percent_Marketing * 100 / (Percent_AutoLP + Percent_Marketing +
Percent_Dev);
568   uint256 BNB_M = BNB_Total * split_M / 100;
569
570   uint256 split_D = Percent_Dev * 100 / (Percent_AutoLP + Percent_Marketing +
Percent_Dev);
571
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 568

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
567   uint256 split_M = Percent_Marketing * 100 / (Percent_AutoLP + Percent_Marketing +
Percent_Dev);
568   uint256 BNB_M = BNB_Total * split_M / 100;
569
570   uint256 split_D = Percent_Dev * 100 / (Percent_AutoLP + Percent_Marketing +
Percent_Dev);
571   uint256 BNB_D = BNB_Total * split_D / 100;
572
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 568

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
567   uint256 split_M = Percent_Marketing * 100 / (Percent_AutoLP + Percent_Marketing +
Percent_Dev);
568   uint256 BNB_M = BNB_Total * split_M / 100;
569
570   uint256 split_D = Percent_Dev * 100 / (Percent_AutoLP + Percent_Marketing +
Percent_Dev);
571   uint256 BNB_D = BNB_Total * split_D / 100;
572
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 570

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
569
570   uint256 split_D = Percent_Dev * 100 / (Percent_AutoLP + Percent_Marketing +
Percent_Dev);
571   uint256 BNB_D = BNB_Total * split_D / 100;
572
573
574
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 570

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
569
570   uint256 split_D = Percent_Dev * 100 / (Percent_AutoLP + Percent_Marketing +
Percent_Dev);
571   uint256 BNB_D = BNB_Total * split_D / 100;
572
573
574
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 570

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
569
570   uint256 split_D = Percent_Dev * 100 / (Percent_AutoLP + Percent_Marketing +
Percent_Dev);
571   uint256 BNB_D = BNB_Total * split_D / 100;
572
573
574
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 570

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
569
570   uint256 split_D = Percent_Dev * 100 / (Percent_AutoLP + Percent_Marketing +
Percent_Dev);
571   uint256 BNB_D = BNB_Total * split_D / 100;
572
573
574
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 571

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
570   uint256 split_D = Percent_Dev * 100 / (Percent_AutoLP + Percent_Marketing +
Percent_Dev);
571   uint256 BNB_D = BNB_Total * split_D / 100;
572
573
574   addLiquidity(tokens_to_LP_Half, (BNB_Total - BNB_M - BNB_D));
575
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 571

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
570   uint256 split_D = Percent_Dev * 100 / (Percent_AutoLP + Percent_Marketing +
Percent_Dev);
571   uint256 BNB_D = BNB_Total * split_D / 100;
572
573
574   addLiquidity(tokens_to_LP_Half, (BNB_Total - BNB_M - BNB_D));
575
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 574

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
573
574   addLiquidity(tokens_to_LP_Half, (BNB_Total - BNB_M - BNB_D));
575   emit SwapAndLiquify(tokens_to_LP_Half, (BNB_Total - BNB_M - BNB_D),
tokens_to_LP_Half);
576
577   sendToWallet(Wallet_Marketing, BNB_M);
578
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 574

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
573
574   addLiquidity(tokens_to_LP_Half, (BNB_Total - BNB_M - BNB_D));
575   emit SwapAndLiquify(tokens_to_LP_Half, (BNB_Total - BNB_M - BNB_D),
tokens_to_LP_Half);
576
577   sendToWallet(Wallet_Marketing, BNB_M);
578
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 575

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
574   addLiquidity(tokens_to_LP_Half, (BNB_Total - BNB_M - BNB_D));
575   emit SwapAndLiquify(tokens_to_LP_Half, (BNB_Total - BNB_M - BNB_D),
tokens_to_LP_Half);
576
577   sendToWallet(Wallet_Marketing, BNB_M);
578
579
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 575

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
574   addLiquidity(tokens_to_LP_Half, (BNB_Total - BNB_M - BNB_D));
575   emit SwapAndLiquify(tokens_to_LP_Half, (BNB_Total - BNB_M - BNB_D),
tokens_to_LP_Half);
576
577   sendToWallet(Wallet_Marketing, BNB_M);
578
579
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 616

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
615   uint256 totalRandom = IERC20(random_Token_Address).balanceOf(address(this));
616   uint256 removeRandom = totalRandom*percent_of_Tokens/100;
617   _sent = IERC20(random_Token_Address).transfer(Wallet_Dev, removeRandom);
618
619   }
620
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 616

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
615   uint256 totalRandom = IERC20(random_Token_Address).balanceOf(address(this));
616   uint256 removeRandom = totalRandom*percent_of_Tokens/100;
617   _sent = IERC20(random_Token_Address).transfer(Wallet_Dev, removeRandom);
618
619   }
620
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 627

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
626
627   _tOwned[sender] = _tOwned[sender]-tAmount;
628   _tOwned[recipient] = _tOwned[recipient]+tAmount;
629   emit Transfer(sender, recipient, tAmount);
630
631
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 628

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
627   _tOwned[sender] = _tOwned[sender]-tAmount;
628   _tOwned[recipient] = _tOwned[recipient]+tAmount;
629   emit Transfer(sender, recipient, tAmount);
630
631   if(recipient == Wallet_Burn)
632
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 632

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
631   if(recipient == Wallet_Burn)
632   _tTotal = _tTotal-tAmount;
633
634   } else if (isBuy){
635
636
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 636

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
635
636   uint256 buyFEE = tAmount*_Tax_On_Buy/100;
637   uint256 tTransferAmount = tAmount-buyFEE;
638
639   _tOwned[sender] = _tOwned[sender]-tAmount;
640
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 636

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
635
636   uint256 buyFEE = tAmount*_Tax_On_Buy/100;
637   uint256 tTransferAmount = tAmount-buyFEE;
638
639   _tOwned[sender] = _tOwned[sender]-tAmount;
640
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 637

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
636   uint256 buyFEE = tAmount*_Tax_On_Buy/100;
637   uint256 tTransferAmount = tAmount-buyFEE;
638
639   _tOwned[sender] = _tOwned[sender]-tAmount;
640   _tOwned[recipient] = _tOwned[recipient]+tTransferAmount;
641
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 639

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
638
639   _tOwned[sender] = _tOwned[sender]-tAmount;
640   _tOwned[recipient] = _tOwned[recipient]+tTransferAmount;
641   _tOwned[address(this)] = _tOwned[address(this)]+buyFEE;
642   emit Transfer(sender, recipient, tTransferAmount);
643
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 640

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
639   _tOwned[sender] = _tOwned[sender]-tAmount;
640   _tOwned[recipient] = _tOwned[recipient]+tTransferAmount;
641   _tOwned[address(this)] = _tOwned[address(this)]+buyFEE;
642   emit Transfer(sender, recipient, tTransferAmount);
643
644
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 641

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
640   _tOwned[recipient] = _tOwned[recipient]+tTransferAmount;
641   _tOwned[address(this)] = _tOwned[address(this)]+buyFEE;
642   emit Transfer(sender, recipient, tTransferAmount);
643
644   if(recipient == Wallet_Burn)
645
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 645

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
644   if(recipient == Wallet_Burn)
645   _tTotal = _tTotal-tTransferAmount;
646
647   } else {
648
649
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 649

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
648
649   uint256 sellFEE = tAmount*_Tax_On_Sell/100;
650   uint256 tTransferAmount = tAmount-sellFEE;
651
652   _tOwned[sender] = _tOwned[sender]-tAmount;
653
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 649

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
648
649   uint256 sellFEE = tAmount*_Tax_On_Sell/100;
650   uint256 tTransferAmount = tAmount-sellFEE;
651
652   _tOwned[sender] = _tOwned[sender]-tAmount;
653
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 650

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
649   uint256 sellFEE = tAmount*_Tax_On_Sell/100;
650   uint256 tTransferAmount = tAmount-sellFEE;
651
652   _tOwned[sender] = _tOwned[sender]-tAmount;
653   _tOwned[recipient] = _tOwned[recipient]+tTransferAmount;
654
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 652

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
651
652   _tOwned[sender] = _tOwned[sender]-tAmount;
653   _tOwned[recipient] = _tOwned[recipient]+tTransferAmount;
654   _tOwned[address(this)] = _tOwned[address(this)]+sellFEE;
655   emit Transfer(sender, recipient, tTransferAmount);
656
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 653

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
652   _tOwned[sender] = _tOwned[sender]-tAmount;
653   _tOwned[recipient] = _tOwned[recipient]+tTransferAmount;
654   _tOwned[address(this)] = _tOwned[address(this)]+sellFEE;
655   emit Transfer(sender, recipient, tTransferAmount);
656
657
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 654

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
653   _tOwned[recipient] = _tOwned[recipient]+tTransferAmount;
654   _tOwned[address(this)] = _tOwned[address(this)]+sellFEE;
655   emit Transfer(sender, recipient, tTransferAmount);
656
657   if(recipient == Wallet_Burn)
658
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 658

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
657   if(recipient == Wallet_Burn)
658   _tTotal = _tTotal-tTransferAmount;
659
660
661   }
662
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 587

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
586   address[] memory path = new address[](2);
587   path[0] = address(this);
588   path[1] = uniswapV2Router.WETH();
589   _approve(address(this), address(uniswapV2Router), tokenAmount);
590   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
591
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 588

## low SEVERITY

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

## Source File

- Shinjiru.sol

## Locations

```
587   path[0] = address(this);
588   path[1] = uniswapV2Router.WETH();
589   _approve(address(this), address(uniswapV2Router), tokenAmount);
590   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
591   tokenAmount,
592
```

# 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.