Ai District

Smart Contract Audit Report

SYSFIXED

05 Feb 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Ai District	AID	Binance Smart Chain

Addresses

Contract address	0x7e37B487a46d4DFbA47fDd7e4A0723f5Ea7D33C2
Contract deployer address	0x59ce9e317407dF204bD423f0A221d359E7b305ae

Project Website

https://www.aidistrict.xyz/

Codebase

https://bscscan.com/address/0x7e37B487a46d4DFbA47fDd7e4A0723f5Ea7D33C2#code

SUMMARY

AI District is the DAO for AI innovation, community-driven exploration and investment in the future of AI. We guide generations X, Y and Z through AI and learn it together.

Contract Summary

Documentation Quality

Ai District 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 Ai District 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 206, 227, 253, 279, 282, 407, 407, 408, 408, 409, 409, 412, 412, 446, 446, 480, 489, 501, 520, 527, 531, 545, 554, 554, 562, 562, 572, 572, 579, 582, 582, 601, 602, 602, 604, 609, 610, 610, 613, 619, 619, 624, 624, 676, 676, 683, 683, 692, 692, 723, 736, 742, 742, 745, 745, 745 and 746.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 16.
- 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 636, 638 and 736.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 545 and 705.

CONCLUSION

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

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

The issues found in the Ai District 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, weak sources of randomness and out of bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	PASS
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
Unprotected Ether Withdrawal	SWC-105	Due to missing or insufficient access controls, malicious parties can withdraw from the contract.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Uninitialized Storage Pointer	SWC-109	Uninitialized local storage variables can point to unexpected storage locations in the contract.	PASS
Assert Violation	SWC-110 SWC-123	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS

DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	PASS
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique ID	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	ISSUE FOUND
Write to Arbitrary Storage Location	SWC-124	The contract is responsible for ensuring that only authorized user or contract accounts may write to sensitive storage locations.	PASS
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS
Insufficient Gas Griefing	SWC-126	Insufficient gas griefing attacks can be performed on contracts which accept data and use it in a sub-call on another contract.	PASS
Arbitrary Jump Function	SWC-127	As Solidity doesnt support pointer arithmetics, it is impossible to change such variable to an arbitrary value.	PASS

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

SMART CONTRACT ANALYSIS

Started	Saturday Feb 04 2023 15:31:58 GMT+0000 (Coordinated Universal Time)
Finished	Sunday Feb 05 2023 08:45:46 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	AIDistrict.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-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 206

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
205   /**
206   * @dev Atomically increases the allowance granted to `spender` by the caller.
207   *
208   * This is an alternative to {approve} that can be used as a mitigation for
209   * problems described in {IBEP20-approve}.
210
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 227

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
226   /**
227   * @dev Atomically decreases the allowance granted to `spender` by the caller.
228   *
229   * This is an alternative to {approve} that can be used as a mitigation for
230   * problems described in {IBEP20-approve}.
231
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 253

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
252   /**
253   * @dev Moves tokens `amount` from `sender` to `recipient`.
254   *
255   * This is internal function is equivalent to {transfer}, and can be used to
256   * e.g. implement automatic token fees, slashing mechanisms, etc.
257
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 279

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
278
279   emit Transfer(sender, recipient, amount);
280   }
281
282   /** This function will be used to generate the total supply
283
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 282

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
281
282   /** This function will be used to generate the total supply
283   * while deploying the contract
284   *
285   * This function can never be called again after deploying contract
286
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 407

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
406   uint256 public tokenLiquidityThreshold = 1e6 * 10**18; // 0.1% of total supply
407   uint256 public maxBuyLimit = 1e8 * 10**18; // 1% of total supply
408   uint256 public maxSellLimit = 1e8 * 10**18; // 1% of total supply
409   uint256 public maxWalletLimit = 1e8 * 10**18; // 1% of total supply
410
411
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 407

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
406   uint256 public tokenLiquidityThreshold = 1e6 * 10**18; // 0.1% of total supply
407   uint256 public maxBuyLimit = 1e8 * 10**18; // 1% of total supply
408   uint256 public maxSellLimit = 1e8 * 10**18; // 1% of total supply
409   uint256 public maxWalletLimit = 1e8 * 10**18; // 1% of total supply
410
411
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 408

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
407   uint256 public maxBuyLimit = 1e8 * 10**18; // 1% of total supply
408   uint256 public maxSellLimit = 1e8 * 10**18; // 1% of total supply
409   uint256 public maxWalletLimit = 1e8 * 10**18; // 1% of total supply
410
411   uint256 public genesis_block;
412
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 408

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
407   uint256 public maxBuyLimit = 1e8 * 10**18; // 1% of total supply
408   uint256 public maxSellLimit = 1e8 * 10**18; // 1% of total supply
409   uint256 public maxWalletLimit = 1e8 * 10**18; // 1% of total supply
410
411   uint256 public genesis_block;
412
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 409

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
408   uint256 public maxSellLimit = 1e8 * 10**18; // 1% of total supply
409   uint256 public maxWalletLimit = 1e8 * 10**18; // 1% of total supply
410
411   uint256 public genesis_block;
412   uint256 private deadline = 3;
413
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 409

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
408   uint256 public maxSellLimit = 1e8 * 10**18; // 1% of total supply
409   uint256 public maxWalletLimit = 1e8 * 10**18; // 1% of total supply
410
411   uint256 public genesis_block;
412   uint256 private deadline = 3;
413
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 412

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
411   uint256 public genesis_block;
412   uint256 private deadline = 3;
413   uint256 private launchtax = 99;
414
415   address public marketingWallet = 0x178ae733d0539D4946B6A5c52b61646E4464830D;
416
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 412

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
411   uint256 public genesis_block;
412   uint256 private deadline = 3;
413   uint256 private launchtax = 99;
414
415   address public marketingWallet = 0x178ae733d0539D4946B6A5c52b61646E4464830D;
416
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 446

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
445
446   IRouter _router = IRouter(0x10ED43C718714eb63d5aA57B78B54704E256024E);
447   // Create a pancake pair for this new token
448   address _pair = IFactory(_router.factory()).createPair(address(this),
_router.WETH());
449
450
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 446

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
445
446   IRouter _router = IRouter(0x10ED43C718714eb63d5aA57B78B54704E256024E);
447   // Create a pancake pair for this new token
448   address _pair = IFactory(_router.factory()).createPair(address(this),
_router.WETH());
449
450
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 480

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
479
480   function increaseAllowance(address spender, uint256 addedValue)
481   public
482   override
483   returns (bool)
484
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 489

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
488
489   function decreaseAllowance(address spender, uint256 subtractedValue)
490   public
491   override
492   returns (bool)
493
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 501

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
500
501   function transfer(address recipient, uint256 amount) public override returns (bool)
{
502   _transfer(msg.sender, recipient, amount);
503   return true;
504   }
505
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 520

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
519   require(balanceOf(recipient) + amount <= maxWalletLimit,"You are exceeding
maxWalletLimit");
520   }
521
522   if (sender != pair && !exemptFee[recipient] && !exemptFee[sender] && !_interlock) {
523   require(amount <= maxSellLimit, "You are exceeding maxSellLimit");
524
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 527

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
526   require(balanceOf(recipient) + amount <= maxWalletLimit,"You are exceeding
maxWalletLimit");
527   }
528
529   if (coolDownEnabled) {
530   uint256 timePassed = block.timestamp - _lastSell[sender];
531
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 531

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
530   uint256 timePassed = block.timestamp - _lastSell[sender];
531   require(timePassed >= coolDownTime, "Cooldown enabled");
532   _lastSell[sender] = block.timestamp;
533   }
534   }
535
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 545

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
544
545   //set fee to zero if fees in contract are handled or exempted
546   if (_interlock || exemptFee[sender] || exemptFee[recipient])
547   fee = 0;
548
549
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 554

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
553   sellTaxes.marketing +
554   sellTaxes.treasury;
555   feesum = feeswap;
556   currentTaxes = sellTaxes;
557   } else if (!useLaunchFee) {
558
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 554

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
553   sellTaxes.marketing +
554   sellTaxes.treasury;
555   feesum = feeswap;
556   currentTaxes = sellTaxes;
557   } else if (!useLaunchFee) {
558
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 562

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
561   taxes.treasury;
562   feesum = feeswap;
563   currentTaxes = taxes;
564   } else if (useLaunchFee) {
565   feeswap = launchtax;
566
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 562

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
561   taxes.treasury;
562   feesum = feeswap;
563   currentTaxes = taxes;
564   } else if (useLaunchFee) {
565   feeswap = launchtax;
566
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 572

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
571   //send fees if threshold has been reached
572   //don't do this on buys, breaks swap
573   if (providingLiquidity && sender != pair) Liquify(feeswap, currentTaxes);
574
575   //rest to recipient
576
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 572

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
571   //send fees if threshold has been reached
572   //don't do this on buys, breaks swap
573   if (providingLiquidity && sender != pair) Liquify(feeswap, currentTaxes);
574
575   //rest to recipient
576
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 579

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
578   //send the fee to the contract
579   if (feeswap > 0) {
580   uint256 feeAmount = (amount * feeswap) / 100;
581   super._transfer(sender, address(this), feeAmount);
582   }
583
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 582

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
581   super._transfer(sender, address(this), feeAmount);
582   }
583
584   }
585   }
586
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 582

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
581   super._transfer(sender, address(this), feeAmount);
582   }
583
584   }
585   }
586
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 601

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
600   uint256 denominator = feeswap * 3;
601   uint256 tokensToAddLiquidityWith = (contractBalance * swapTaxes.liquidity) /
denominator;
602   uint256 toSwap = contractBalance - tokensToAddLiquidityWith;
603
604   uint256 initialBalance = address(this).balance;
605
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 602

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
601   uint256 tokensToAddLiquidityWith = (contractBalance * swapTaxes.liquidity) /
denominator;
602   uint256 toSwap = contractBalance - tokensToAddLiquidityWith;
603
604   uint256 initialBalance = address(this).balance;
605
606
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 602

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
601   uint256 tokensToAddLiquidityWith = (contractBalance * swapTaxes.liquidity) /
denominator;
602   uint256 toSwap = contractBalance - tokensToAddLiquidityWith;
603
604   uint256 initialBalance = address(this).balance;
605
606
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 604

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
603
604   uint256 initialBalance = address(this).balance;
605
606   swapTokensForETH(toSwap);
607
608
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 609

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
608   uint256 deltaBalance = address(this).balance - initialBalance;
609   uint256 unitBalance = deltaBalance / (denominator - swapTaxes.liquidity);
610   uint256 ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity;
611
612   if (ethToAddLiquidityWith > 0) {
613
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 610

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
609   uint256 unitBalance = deltaBalance / (denominator - swapTaxes.liquidity);
610   uint256 ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity;
611
612   if (ethToAddLiquidityWith > 0) {
613   // Add liquidity to pancake
614
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 610

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
609   uint256 unitBalance = deltaBalance / (denominator - swapTaxes.liquidity);
610   uint256 ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity;
611
612   if (ethToAddLiquidityWith > 0) {
613   // Add liquidity to pancake
614
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 613

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
612   if (ethToAddLiquidityWith > 0) {
613   // Add liquidity to pancake
614   addLiquidity(tokensToAddLiquidityWith, ethToAddLiquidityWith);
615   }
616
617
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 619

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
618   if (marketingAmt > 0) {
619   payable(marketingWallet).sendValue(marketingAmt);
620   }
621
622   uint256 treasuryAmt = unitBalance * 3 * swapTaxes.treasury;
623
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 619

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
618   if (marketingAmt > 0) {
619   payable(marketingWallet).sendValue(marketingAmt);
620   }
621
622   uint256 treasuryAmt = unitBalance * 3 * swapTaxes.treasury;
623
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 624

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
623   if (treasuryAmt > 0) {
624   payable(treasuryWallet).sendValue(treasuryAmt);
625   }
626
627   }
628
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 624

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
623   if (treasuryAmt > 0) {
624   payable(treasuryWallet).sendValue(treasuryAmt);
625   }
626
627   }
628
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 676

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
675   uint256 _marketing,
676   uint256 _treasury,
677   uint256 _liquidity
678   ) external onlyOwner {
679   taxes = Taxes(_marketing, _treasury, _liquidity);
680
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 676

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
675   uint256 _marketing,
676   uint256 _treasury,
677   uint256 _liquidity
678   ) external onlyOwner {
679   taxes = Taxes(_marketing, _treasury, _liquidity);
680
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 683

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
682
683   function SetSellTaxes(
684   uint256 _marketing,
685   uint256 _treasury,
686   uint256 _liquidity
687
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 683

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
682
683   function SetSellTaxes(
684   uint256 _marketing,
685   uint256 _treasury,
686   uint256 _liquidity
687
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 692

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
691
692   function updateRouterAndPair(address newRouter, address newPair) external onlyOwner
{
693   router = IRouter(newRouter);
694   pair = newPair;
695   }
696
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 692

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
691
692   function updateRouterAndPair(address newRouter, address newPair) external onlyOwner
{
693   router = IRouter(newRouter);
694   pair = newPair;
695   }
696
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 723

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
722   coolDownEnabled = state;
723   require(time <= 300, "cooldown timer cannot exceed 5 minutes");
724   }
725
726   function updateExemptFee(address _address, bool state) external onlyOwner {
727
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 736

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
735
736   function updateMaxTxLimit(uint256 maxBuy, uint256 maxSell, uint256 maxWallet)
external onlyOwner {
737   require(maxBuy >= 1e6, "Cannot set max buy amount lower than 0.1%");
738   require(maxSell >= 1e6, "Cannot set max sell amount lower than 0.1%");
739   require(maxWallet >= 1e6, "Cannot set max wallet amount lower than 1%");
740
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 742

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
741   maxSellLimit = maxSell * 10**decimals();
742   maxWalletLimit = maxWallet * 10**decimals();
743   }
744
745   function rescueBNB(uint256 weiAmount) external onlyOwner {
746
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 742

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
741   maxSellLimit = maxSell * 10**decimals();
742   maxWalletLimit = maxWallet * 10**decimals();
743   }
744
745   function rescueBNB(uint256 weiAmount) external onlyOwner {
746
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 745

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
744
745   function rescueBNB(uint256 weiAmount) external onlyOwner {
746   payable(owner()).transfer(weiAmount);
747   }
748
749
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 745

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
744
745   function rescueBNB(uint256 weiAmount) external onlyOwner {
746   payable(owner()).transfer(weiAmount);
747   }
748
749
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 745

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
744
745   function rescueBNB(uint256 weiAmount) external onlyOwner {
746   payable(owner()).transfer(weiAmount);
747   }
748
749
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 746

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
745   function rescueBNB(uint256 weiAmount) external onlyOwner {
746   payable(owner()).transfer(weiAmount);
747   }
748
749   function rescueBSC20(address tokenAdd, uint256 amount) external onlyOwner {
750
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 16

low SEVERITY

The current pragma Solidity directive is ""^0.8.18"". 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

- AIDistrict.sol

Locations

```
15   abstract contract Context {
16   function _msgSender() internal view virtual returns (address) {
17   return msg.sender;
18   }
19
20
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 636

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
635
636   _approve(address(this), address(router), tokenAmount);
637
638   // make the swap
639   router.swapExactTokensForETHSupportingFeeOnTransferTokens(
640
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 638

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
637
638   // make the swap
639   router.swapExactTokensForETHSupportingFeeOnTransferTokens(
640   tokenAmount,
641   0,
642
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 736

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
735
736   function updateMaxTxLimit(uint256 maxBuy, uint256 maxSell, uint256 maxWallet)
external onlyOwner {
737   require(maxBuy >= 1e6, "Cannot set max buy amount lower than 0.1%");
738   require(maxSell >= 1e6, "Cannot set max sell amount lower than 0.1%");
739   require(maxWallet >= 1e6, "Cannot set max wallet amount lower than 1%");
740
```

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

LINE 545

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
544
545   //set fee to zero if fees in contract are handled or exempted
546   if (_interlock || exemptFee[sender] || exemptFee[recipient])
547   fee = 0;
548
549
```

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

LINE 705

low SEVERITY

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

Source File

- AIDistrict.sol

Locations

```
704   function updatedeadline(uint256 _deadline) external onlyOwner {
705   require(!tradingEnabled, "Can't change when trading has started");
706   require(_deadline < 5,"Deadline should be less than 5 Blocks");
707   deadline = _deadline;
708   }
709
```

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.