Archon

Smart Contract Audit Report

SYSFIXED

19 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Archon	$ARCH	Binance Smart Chain

Addresses

Contract address	0x8A9a97591503515538CD167E0cF05BE7C004628C
Contract deployer address	0x8E31bA50931E753D1359cfC4f4bf4F10Db0353Ea

Project Website

https://archon.live/

Codebase

https://bscscan.com/address/0x8A9a97591503515538CD167E0cF05BE7C004628C#code

SUMMARY

Archon MMORPG is the World's First Open World Third Person P2E MMORPG with Next Gen Graphics. Archon MMORPG Game Demo V1 is Live on the Official Archon Website. Download, Install and Run it Up on the PC. PS5 Developments are ongoing. Multiple Archon MMORPG Series are going Live. Archon has KYC+ Audit| 0% Tax| Gate.io Onboarding| Certik Incoming| Binance AMA and more. Archon Staking Platform 2.0 with no Lock Duration and Platform charge goes Live for the Holders after the Launch

Contract Summary

Documentation Quality

Archon 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 Archon with the discovery of several low issues.

Test Coverage

Test coverage of the project is 100% (Through Codebase)

Audit Findings Summary

- SWC-100 SWC-108 | Explicitly define visibility for all state variables on lines 196, 199, 275 and 287.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 280, 280, 329, 329, 330, 330, 382, 382, 455, 464, 472, 498, 508, 508, 556, 556, 569, 569, 576, 577, 579, 579, 626, 627, 651, 664, 664, 666, 666, 667, 667, 677, 677, 677, 680, 689, 689, 689, 701, 705, 705, 705, 740, 740, 740, 740, 740, 740, 740, 740, 740, 740, 740, 740 and 740.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 85.
- 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 684, 685, 740, 740, 740, 740, 740, 740, 740 and 740.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 608.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 726.

CONCLUSION

We have audited the Archon project released on January 2023 to discover issues and identify potential security vulnerabilities in ArchonProject. 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 Archon smart contract code do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are some arithmetic operation issues, a floating pragma is set, a state variable visibility is not set, weak sources of randomness, tx.origin as a part of authorization control and out of bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	ISSUE FOUND
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.	ISSUE FOUND
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	Wednesday Jan 18 2023 05:58:36 GMT+0000 (Coordinated Universal Time)
Finished	Thursday Jan 19 2023 18:24:50 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Archon.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-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	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-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-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-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 280

low SEVERITY

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

Source File

- Archon.sol

Locations

```
279
280   uint256 public swapThreshold = (_tTotal * 10) / 10000;
281   uint256 public swapAmount = (_tTotal * 10) / 10000;
282   uint256 public swapInterval = 0;
283   uint256 public lastSwap;
284
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 280

low SEVERITY

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

Source File

- Archon.sol

Locations

```
279
280   uint256 public swapThreshold = (_tTotal * 10) / 10000;
281   uint256 public swapAmount = (_tTotal * 10) / 10000;
282   uint256 public swapInterval = 0;
283   uint256 public lastSwap;
284
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 329

low SEVERITY

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

Source File

- Archon.sol

Locations

```
328
329   _approve(msg.sender, currentRouter, type(uint256).max);
330   _approve(address(this), currentRouter, type(uint256).max);
331
332   _isExcludedFromFees[owner()] = true;
333
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 329

low SEVERITY

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

Source File

- Archon.sol

Locations

```
328
329   _approve(msg.sender, currentRouter, type(uint256).max);
330   _approve(address(this), currentRouter, type(uint256).max);
331
332   _isExcludedFromFees[owner()] = true;
333
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 330

low SEVERITY

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

Source File

- Archon.sol

Locations

```
329   _approve(msg.sender, currentRouter, type(uint256).max);
330   _approve(address(this), currentRouter, type(uint256).max);
331
332   _isExcludedFromFees[owner()] = true;
333   _isExcludedFromFees[address(this)] = true;
334
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 330

low SEVERITY

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

Source File

- Archon.sol

Locations

```
329   _approve(msg.sender, currentRouter, type(uint256).max);
330   _approve(address(this), currentRouter, type(uint256).max);
331
332   _isExcludedFromFees[owner()] = true;
333   _isExcludedFromFees[address(this)] = true;
334
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 382

low SEVERITY

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

Source File

- Archon.sol

Locations

```
381   function decimals() external view override returns (uint8) { return _decimals; }
382   function symbol() external pure override returns (string memory) { return _symbol;
}
383   function name() external pure override returns (string memory) { return _name; }
384   function getOwner() external view override returns (address) { return owner(); }
385   function allowance(address holder, address spender) external view override returns
(uint256) { return _allowances[holder][spender]; }
386
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 382

low SEVERITY

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

Source File

- Archon.sol

Locations

```
381   function decimals() external view override returns (uint8) { return _decimals; }
382   function symbol() external pure override returns (string memory) { return _symbol;
}
383   function name() external pure override returns (string memory) { return _name; }
384   function getOwner() external view override returns (address) { return owner(); }
385   function allowance(address holder, address spender) external view override returns
(uint256) { return _allowances[holder][spender]; }
386
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 455

low SEVERITY

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

Source File

- Archon.sol

Locations

```
454   timeSinceLastPair = block.timestamp;
455   antiSnipe.setLpPair(pair, true);
456   }
457   }
458
459
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 464

low SEVERITY

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

Source File

- Archon.sol

Locations

```
463
464   function getCirculatingSupply() public view returns (uint256) {
465   return (_tTotal - (balanceOf(DEAD) + balanceOf(address(0))));
466   }
467
468
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 472

low SEVERITY

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

Source File

- Archon.sol

Locations

```
471
472   function setExcludedFromFees(address account, bool enabled) public onlyOwner {
473   _isExcludedFromFees[account] = enabled;
474   }
475   //Initialize the Anti Snipe measures.
476
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 498

low SEVERITY

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

Source File

- Archon.sol

Locations

```
497
498   function setProtectionSettings(bool _antiSnipe, bool _antiGas, bool _antiBlock,
bool _algo) external onlyOwner {
499   antiSnipe.setProtections(_antiSnipe, _antiGas, _antiBlock, _algo);
500   }
501
502
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 508

low SEVERITY

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

Source File

- Archon.sol

Locations

```
507   function setTaxes(uint16 buyFee, uint16 sellFee, uint16 transferFee) external
onlyOwner {
508   require(buyFee <= staticVals.maxBuyTaxes
509   && sellFee <=staticVals. maxSellTaxes
510   && transferFee <= staticVals.maxTransferTaxes,
511   "Cannot exceed maximums of 25%.");
512
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 508

low SEVERITY

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

Source File

- Archon.sol

Locations

```
507   function setTaxes(uint16 buyFee, uint16 sellFee, uint16 transferFee) external
onlyOwner {
508   require(buyFee <= staticVals.maxBuyTaxes
509   && sellFee <=staticVals. maxSellTaxes
510   && transferFee <= staticVals.maxTransferTaxes,
511   "Cannot exceed maximums of 25%.");
512
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 556

low SEVERITY

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

Source File

- Archon.sol

Locations

```
555   _liquidityHolders[presale] = true;
556   presaleAddresses[presale] = true;
557   setExcludedFromFees(presale, true);
558   } else {
559   _liquidityHolders[router] = true;
560
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 556

low SEVERITY

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

Source File

- Archon.sol

Locations

```
555   _liquidityHolders[presale] = true;
556   presaleAddresses[presale] = true;
557   setExcludedFromFees(presale, true);
558   } else {
559   _liquidityHolders[router] = true;
560
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 569

low SEVERITY

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

Source File

- Archon.sol

Locations

```
568   function _hasLimits(address from, address to) private view returns (bool) {
569   return from != owner()
570   && to != owner()
571   && tx.origin != owner()
572   && !_liquidityHolders[to]
573
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 569

low SEVERITY

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

Source File

- Archon.sol

Locations

```
568   function _hasLimits(address from, address to) private view returns (bool) {
569   return from != owner()
570   && to != owner()
571   && tx.origin != owner()
572   && !_liquidityHolders[to]
573
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 576

low SEVERITY

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

Source File

- Archon.sol

Locations

```
575   && to != address(0)
576   && from != address(this);
577   }
578
579   function _transfer(address from, address to, uint256 amount) internal returns
(bool) {
580
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 577

low SEVERITY

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

Source File

- Archon.sol

Locations

```
576   && from != address(this);
577   }
578
579   function _transfer(address from, address to, uint256 amount) internal returns
(bool) {
580   require(from != address(0), "ERC20: transfer from the zero address");
581
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 579

low SEVERITY

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

Source File

- Archon.sol

Locations

```
578
579   function _transfer(address from, address to, uint256 amount) internal returns
(bool) {
580   require(from != address(0), "ERC20: transfer from the zero address");
581   require(to != address(0), "ERC20: transfer to the zero address");
582   require(amount > 0, "Transfer amount must be greater than zero");
583
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 579

low SEVERITY

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

Source File

- Archon.sol

Locations

```
578
579   function _transfer(address from, address to, uint256 amount) internal returns
(bool) {
580   require(from != address(0), "ERC20: transfer from the zero address");
581   require(to != address(0), "ERC20: transfer to the zero address");
582   require(amount > 0, "Transfer amount must be greater than zero");
583
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 626

low SEVERITY

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

Source File

- Archon.sol

Locations

```
625   }
626
627   function contractSwap(uint256 contractTokenBalance) private lockTheSwap {
628   if (_ratios.total == 0)
629   return;
630
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 627

low SEVERITY

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

Source File

- Archon.sol

Locations

```
626
627   function contractSwap(uint256 contractTokenBalance) private lockTheSwap {
628   if (_ratios.total == 0)
629   return;
630
631
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 651

low SEVERITY

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

Source File

- Archon.sol

Locations

```
650
651   uint256 liquidityBalance = ((address(this).balance * _ratios.liquidity) /
_ratios.total) / 2;
652
653   if (toLiquify > 0) {
654   dexRouter.addLiquidityETH{value: liquidityBalance}(
655
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 664

low SEVERITY

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

Source File

- Archon.sol

Locations

```
663   }
664   if (address(this).balance > 0 && _ratios.total - _ratios.liquidity > 0) {
665   uint256 amountBNB = address(this).balance;
666   _taxWallets.development.transfer((amountBNB * _ratios.development) / (_ratios.total
- _ratios.liquidity));
667   _taxWallets.marketing.transfer(address(this).balance);
668
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 664

low SEVERITY

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

Source File

- Archon.sol

Locations

```
663   }
664   if (address(this).balance > 0 && _ratios.total - _ratios.liquidity > 0) {
665   uint256 amountBNB = address(this).balance;
666   _taxWallets.development.transfer((amountBNB * _ratios.development) / (_ratios.total
- _ratios.liquidity));
667   _taxWallets.marketing.transfer(address(this).balance);
668
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 666

low SEVERITY

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

Source File

- Archon.sol

Locations

```
665   uint256 amountBNB = address(this).balance;
666   _taxWallets.development.transfer((amountBNB * _ratios.development) / (_ratios.total
- _ratios.liquidity));
667   _taxWallets.marketing.transfer(address(this).balance);
668   }
669   }
670
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 666

low SEVERITY

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

Source File

- Archon.sol

Locations

```
665   uint256 amountBNB = address(this).balance;
666   _taxWallets.development.transfer((amountBNB * _ratios.development) / (_ratios.total
- _ratios.liquidity));
667   _taxWallets.marketing.transfer(address(this).balance);
668   }
669   }
670
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 667

low SEVERITY

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

Source File

- Archon.sol

Locations

```
666   _taxWallets.development.transfer((amountBNB * _ratios.development) / (_ratios.total
- _ratios.liquidity));
667   _taxWallets.marketing.transfer(address(this).balance);
668   }
669   }
670
671
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 667

low SEVERITY

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

Source File

- Archon.sol

Locations

```
666   _taxWallets.development.transfer((amountBNB * _ratios.development) / (_ratios.total
- _ratios.liquidity));
667   _taxWallets.marketing.transfer(address(this).balance);
668   }
669   }
670
671
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 677

low SEVERITY

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

Source File

- Archon.sol

Locations

```
676   if(address(antiSnipe) == address(0)){
677   antiSnipe = AntiSnipe(address(this));
678   }
679   contractSwapEnabled = true;
680   emit ContractSwapEnabledUpdated(true);
681
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 677

low SEVERITY

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

Source File

- Archon.sol

Locations

```
676   if(address(antiSnipe) == address(0)){
677   antiSnipe = AntiSnipe(address(this));
678   }
679   contractSwapEnabled = true;
680   emit ContractSwapEnabledUpdated(true);
681
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 677

low SEVERITY

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

Source File

- Archon.sol

Locations

```
676   if(address(antiSnipe) == address(0)){
677   antiSnipe = AntiSnipe(address(this));
678   }
679   contractSwapEnabled = true;
680   emit ContractSwapEnabledUpdated(true);
681
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 680

low SEVERITY

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

Source File

- Archon.sol

Locations

```
679   contractSwapEnabled = true;
680   emit ContractSwapEnabledUpdated(true);
681   }
682   }
683   //Enable Trading
684
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 689

low SEVERITY

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

Source File

- Archon.sol

Locations

```
688   }
689   try antiSnipe.setLaunch(lpPair, uint32(block.number), uint64(block.timestamp),
_decimals) {} catch {}
690   tradingEnabled = true;
691   }
692
693
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 689

low SEVERITY

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

Source File

- Archon.sol

Locations

```
688   }
689   try antiSnipe.setLaunch(lpPair, uint32(block.number), uint64(block.timestamp),
_decimals) {} catch {}
690   tradingEnabled = true;
691   }
692
693
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 689

low SEVERITY

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

Source File

- Archon.sol

Locations

```
688   }
689   try antiSnipe.setLaunch(lpPair, uint32(block.number), uint64(block.timestamp),
_decimals) {} catch {}
690   tradingEnabled = true;
691   }
692
693
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 701

low SEVERITY

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

Source File

- Archon.sol

Locations

```
700   require(balanceOf(msg.sender) >= amounts[i]);
701   _transfer(msg.sender, accounts[i], amounts[i]*10**_decimals);
702   }
703   }
704
705
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 705

low SEVERITY

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

Source File

- Archon.sol

Locations

```
704
705   function multiSendPercents(address[] memory accounts, uint256[] memory percents,
uint256[] memory divisors) external {
706   require(accounts.length == percents.length && percents.length == divisors.length,
"Lengths do not match.");
707   for (uint8 i = 0; i < accounts.length; i++) {
708   require(balanceOf(msg.sender) >= (_tTotal * percents[i]) / divisors[i]);
709
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 705

low SEVERITY

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

Source File

- Archon.sol

Locations

```
704
705   function multiSendPercents(address[] memory accounts, uint256[] memory percents,
uint256[] memory divisors) external {
706   require(accounts.length == percents.length && percents.length == divisors.length,
"Lengths do not match.");
707   for (uint8 i = 0; i < accounts.length; i++) {
708   require(balanceOf(msg.sender) >= (_tTotal * percents[i]) / divisors[i]);
709
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 705

low SEVERITY

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

Source File

- Archon.sol

Locations

```
704
705   function multiSendPercents(address[] memory accounts, uint256[] memory percents,
uint256[] memory divisors) external {
706   require(accounts.length == percents.length && percents.length == divisors.length,
"Lengths do not match.");
707   for (uint8 i = 0; i < accounts.length; i++) {
708   require(balanceOf(msg.sender) >= (_tTotal * percents[i]) / divisors[i]);
709
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 85

low SEVERITY

The current pragma Solidity directive is ""^0.6.0<0.9.0"". 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

- Archon.sol

Locations

```
84   *
85   * This value changes when {approve} or {transferFrom} are called.
86   */
87   function allowance(address _owner, address spender) external view returns (uint256);
88
89
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 196

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "lpPairs" is internal. Other possible visibility settings are public and private.

Source File

- Archon.sol

Locations

```
195   address public officialContractAddress;
196   mapping (address => bool) lpPairs;
197   uint256 private timeSinceLastPair = 0;
198   mapping (address => mapping (address => uint256)) private _allowances;
199   IERC20 token;
200
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 199

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "token" is internal. Other possible visibility settings are public and private.

Source File

- Archon.sol

Locations

```
198   mapping (address => mapping (address => uint256)) private _allowances;
199   IERC20 token;
200
201   mapping (address => bool) private _isExcludedFromFees;
202   mapping (address => bool) private _isExcluded;
203
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 275

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "inSwap" is internal. Other possible visibility settings are public and private.

Source File

- Archon.sol

Locations

```
274
275   bool inSwap;
276   bool public contractSwapEnabled = false;
277
278   uint256 private _maxTxAmountPercent = 5;
279
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 287

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "antiSnipe" is internal. Other possible visibility settings are public and private.

Source File

- Archon.sol

Locations

```
286   bool public _hasLiqBeenAdded = false;
287   AntiSnipe antiSnipe;
288
289   event OwnershipTransferred(address indexed previousOwner, address indexed
newOwner);
290   event ContractSwapEnabledUpdated(bool enabled);
291
```

SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 608

low SEVERITY

The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead.

Source File

- Archon.sol

Locations

```
607   if (contractTokenBalance >= swapThreshold && lastSwap + swapInterval <
block.timestamp) {
608   if(contractTokenBalance >= swapAmount) { contractTokenBalance = swapAmount; }
609   contractSwap(contractTokenBalance);
610   lastSwap = block.timestamp;
611   }
612
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 684

low SEVERITY

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

Source File

- Archon.sol

Locations

```
683   //Enable Trading
684   function enableTrading() public onlyOwner {
685   require(!tradingEnabled, "Trading already enabled!");
686   if(address(antiSnipe) == address(0)){
687   antiSnipe = AntiSnipe(address(this));
688
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 685

low SEVERITY

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

Source File

- Archon.sol

Locations

```
684   function enableTrading() public onlyOwner {
685   require(!tradingEnabled, "Trading already enabled!");
686   if(address(antiSnipe) == address(0)){
687   antiSnipe = AntiSnipe(address(this));
688   }
689
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 740

low SEVERITY

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

Source File

- Archon.sol

Locations

```
739
740   }
741
```

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

LINE 726

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

- Archon.sol

Locations

```
725   _tOwned[address(this)] += feeAmount;
726   emit Transfer(from, address(this), feeAmount);
727
728   return amount - feeAmount;
729   }
730
```

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.