



Ground Zero Smart Contract Audit Report

TABLE OF CONTENTS

[Audited Details](#)

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

[Summary](#)

- Contract Summary
- Audit Findings Summary
- Vulnerabilities Summary

[Conclusion](#)

[Audit Results](#)

[Smart Contract Analysis](#)

- Detected Vulnerabilities

[Disclaimer](#)

[About Us](#)

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Ground Zero	GZT	Binance Smart Chain

Addresses

Contract address	0xA563fdA8864104f933Cf91c6Ca5c6e0a3Dc09b79
Contract deployer address	0x93D87BdFf0c034CCF454A600eAA7d909993426D9

Project Website

<https://www.groundzerobsc.com/>

Codebase

<https://bscscan.com/address/0xA563fdA8864104f933Cf91c6Ca5c6e0a3Dc09b79#code>

SUMMARY

Are you ready for what's coming? The Ground Zero token has a 93% security rating by Contract Wolf and provides protection against global threats, including biological attacks, nuclear bombs, and technological disruptions. With advanced technology including Faraday Cage and Integrated Zeneth IA, invest now in the safer future with the Ground Zero token. The revolution against the existential threat of humanity has arrived.

| Contract Summary

Documentation Quality

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

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

Code Quality

The Overall quality of the basecode is standard.

- Standard solidity basecode and rules are already followed by Ground Zero 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 193, 215, 240, 271, 272, 291, 292, 314, 315, 453, 453, 454, 454, 492, 492, 530, 540, 551, 578, 588, 593, 606, 615, 615, 622, 622, 632, 632, 639, 643, 643, 658, 659, 659, 661, 667, 668, 668, 669, 676, 676, 681, 681, 731, 731, 740, 740, 749, 749, 782, 792, 799, 799, 810 and 820.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 6.
- 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 694, 695, 783, 793 and 821.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 606 and 761.

CONCLUSION

We have audited the Ground Zero project released on February 2023 to discover issues and identify potential security vulnerabilities in NamaFile 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 NamaFile 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 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. We Recommend Don't use any of those environment variables as sources of randomness and being aware that the use of these variables introduces a certain level of trust in miners.

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 grieving attacks can be performed on contracts which accept data and use it in a sub-call on another contract.	PASS
Arbitrary Jump Function	SWC-127	As Solidity doesnt support pointer arithmetics, it is impossible to change such variable to an arbitrary value.	PASS

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

SMART CONTRACT ANALYSIS

Started	Saturday Feb 04 2023 07:13:19 GMT+0000 (Coordinated Universal Time)
Finished	Sunday Feb 05 2023 05:15:35 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	GroundZeroTokenIA.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-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 193

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
192     require(currentAllowance >= amount, "ERC20: transfer amount exceeds allowance");
193     _approve(sender, _msgSender(), currentAllowance - amount);
194
195     return true;
196 }
197
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 215

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
214 {  
215   _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);  
216   return true;  
217 }  
218  
219
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 240

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
239     require(currentAllowance >= subtractedValue, "ERC20: decreased allowance below
zero");
240     _approve(_msgSender(), spender, currentAllowance - subtractedValue);
241
242     return true;
243 }
244
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 271

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
270     require(senderBalance >= amount, "ERC20: transfer amount exceeds balance");
271     _balances[sender] = senderBalance - amount;
272     _balances[recipient] += amount;
273
274     emit Transfer(sender, recipient, amount);
275
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 272

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
271  _balances[sender] = senderBalance - amount;  
272  _balances[recipient] += amount;  
273  
274  emit Transfer(sender, recipient, amount);  
275  }  
276
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 291

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
290
291  _totalSupply += amount;
292  _balances[account] += amount;
293  emit Transfer(address(0), account, amount);
294  }
295
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 292

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
291 _totalSupply += amount;  
292 _balances[account] += amount;  
293 emit Transfer(address(0), account, amount);  
294 }  
295  
296
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 314

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
313     require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
314     _balances[account] = accountBalance - amount;
315     _totalSupply -= amount;
316
317     emit Transfer(account, address(0), amount);
318
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 315

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
314  _balances[account] = accountBalance - amount;  
315  _totalSupply -= amount;  
316  
317  emit Transfer(account, address(0), amount);  
318  }  
319
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 453

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
452
453 uint256 public tokenLiquidityThreshold = 71000000 * 12**decimals();
454 uint256 public maxWalletLimit = 355000000 * 12**decimals();
455
456 uint256 public genesis_block;
457
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 453

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
452
453 uint256 public tokenLiquidityThreshold = 71000000 * 12**decimals();
454 uint256 public maxWalletLimit = 355000000 * 12**decimals();
455
456 uint256 public genesis_block;
457
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 454

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
453 uint256 public tokenLiquidityThreshold = 71000000 * 12**decimals();
454 uint256 public maxWalletLimit = 355000000 * 12**decimals();
455
456 uint256 public genesis_block;
457 uint256 private deadline = 3;
458
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 454

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
453 uint256 public tokenLiquidityThreshold = 71000000 * 12**decimals();
454 uint256 public maxWalletLimit = 355000000 * 12**decimals();
455
456 uint256 public genesis_block;
457 uint256 private deadline = 3;
458
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 492

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
491  constructor(address routerAdd, address serviceFeeReceiver) ERC20("Ground Zero",  
"GZT") payable {  
492  _tokengeneration(msg.sender, 71000000000 * 10**decimals());  
493  exemptFee[msg.sender] = true;  
494  
495  
496
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 492

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
491 constructor(address routerAdd, address serviceFeeReceiver) ERC20("Ground Zero",  
"GZT") payable {  
492 _tokengeneration(msg.sender, 71000000000 * 10**decimals());  
493 exemptFee[msg.sender] = true;  
494  
495  
496
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 530

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
529     require(currentAllowance >= amount, "ERC20: transfer amount exceeds allowance");
530     _approve(sender, _msgSender(), currentAllowance - amount);
531
532     return true;
533 }
534
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 540

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
539 {  
540   _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);  
541   return true;  
542 }  
543  
544
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 551

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
550     require(currentAllowance >= subtractedValue, "ERC20: decreased allowance below
551     _approve(_msgSender(), spender, currentAllowance - subtractedValue);
552
553     return true;
554 }
555
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 578

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
577     require(  
578         balanceOf(recipient) + amount <= maxWalletLimit,  
579         "You are exceeding maxWalletLimit"  
580     );  
581 }  
582
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 588

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
587     require(  
588         balanceOf(recipient) + amount <= maxWalletLimit,  
589         "You are exceeding maxWalletLimit"  
590     );  
591 }  
592
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 593

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
592     if (cooldownEnabled) {  
593         uint256 timePassed = block.timestamp - _lastSell[sender];  
594         require(timePassed >= cooldownTime, "Cooldown enabled");  
595         _lastSell[sender] = block.timestamp;  
596     }  
597
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 606

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
605     !exemptFee[recipient] &&  
606     block.number < genesis_block + deadline;  
607  
608     //set fee to zero if fees in contract are handled or exempted  
609     if (_liquidityMutex || exemptFee[sender] || exemptFee[recipient])  
610
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 615

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
614 feeswap =  
615 sellTaxes.liquidity +  
616 sellTaxes.marketing +  
617 sellTaxes.dev;  
618 feesum = feeswap;  
619
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 615

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
614 feeswap =  
615 sellTaxes.liquidity +  
616 sellTaxes.marketing +  
617 sellTaxes.dev;  
618 feesum = feeswap;  
619
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 622

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
621 feeswap =  
622 taxes.liquidity +  
623 taxes.marketing +  
624 taxes.dev;  
625 feesum = feeswap;  
626
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 622

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
621 feeswap =  
622 taxes.liquidity +  
623 taxes.marketing +  
624 taxes.dev;  
625 feesum = feeswap;  
626
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 632

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
631
632 fee = (amount * feesum) / 100;
633
634 //send fees if threshold has been reached
635 //don't do this on buys, breaks swap
636
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 632

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
631
632 fee = (amount * feesum) / 100;
633
634 //send fees if threshold has been reached
635 //don't do this on buys, breaks swap
636
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 639

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
638 //rest to recipient
639 super._transfer(sender, recipient, amount - fee);
640 if (fee > 0) {
641 //send the fee to the contract
642 if (feeswap > 0) {
643
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 643

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
642  if (feeswap > 0) {  
643    uint256 feeAmount = (amount * feeswap) / 100;  
644    super._transfer(sender, address(this), feeAmount);  
645  }  
646  
647
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 643

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
642  if (feeswap > 0) {  
643      uint256 feeAmount = (amount * feeswap) / 100;  
644      super._transfer(sender, address(this), feeAmount);  
645  }  
646  
647
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 658

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
657 // Split the contract balance into halves
658 uint256 denominator = feeswap * 2;
659 uint256 tokensToAddLiquidityWith = (contractBalance * swapTaxes.liquidity) /
660 denominator;
661 uint256 toSwap = contractBalance - tokensToAddLiquidityWith;
662
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 659

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
658 uint256 denominator = feeswap * 2;  
659 uint256 tokensToAddLiquidityWith = (contractBalance * swapTaxes.liquidity) /  
660 denominator;  
661 uint256 toSwap = contractBalance - tokensToAddLiquidityWith;  
662  
663
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 659

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
658 uint256 denominator = feeswap * 2;  
659 uint256 tokensToAddLiquidityWith = (contractBalance * swapTaxes.liquidity) /  
660 denominator;  
661 uint256 toSwap = contractBalance - tokensToAddLiquidityWith;  
662  
663
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 661

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
660     denominator;  
661     uint256 toSwap = contractBalance - tokensToAddLiquidityWith;  
662  
663     uint256 initialBalance = address(this).balance;  
664  
665
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 667

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
666
667  uint256 deltaBalance = address(this).balance - initialBalance;
668  uint256 unitBalance = deltaBalance / (denominator - swapTaxes.liquidity);
669  uint256 ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity;
670
671
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 668

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
667 uint256 deltaBalance = address(this).balance - initialBalance;
668 uint256 unitBalance = deltaBalance / (denominator - swapTaxes.liquidity);
669 uint256 ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity;
670
671 if (ethToAddLiquidityWith > 0) {
672
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 668

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
667 uint256 deltaBalance = address(this).balance - initialBalance;
668 uint256 unitBalance = deltaBalance / (denominator - swapTaxes.liquidity);
669 uint256 ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity;
670
671 if (ethToAddLiquidityWith > 0) {
672
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 669

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
668 uint256 unitBalance = deltaBalance / (denominator - swapTaxes.liquidity);
669 uint256 ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity;
670
671 if (ethToAddLiquidityWith > 0) {
672     // Add liquidity to uniswap
673 }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 676

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
675
676  uint256 marketingAmt = unitBalance * 2 * swapTaxes.marketing;
677  if (marketingAmt > 0) {
678    payable(marketingWallet).sendValue(marketingAmt);
679  }
680
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 676

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
675
676  uint256 marketingAmt = unitBalance * 2 * swapTaxes.marketing;
677  if (marketingAmt > 0) {
678    payable(marketingWallet).sendValue(marketingAmt);
679  }
680
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 681

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
680
681  uint256 devAmt = unitBalance * 2 * swapTaxes.dev;
682  if (devAmt > 0) {
683    payable(devWallet).sendValue(devAmt);
684  }
685
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 681

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
680
681  uint256 devAmt = unitBalance * 2 * swapTaxes.dev;
682  if (devAmt > 0) {
683    payable(devWallet).sendValue(devAmt);
684  }
685
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 731

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
730 //update the treshhold
731 tokenLiquidityThreshold = new_amount * 12**decimals();
732 }
733
734 function UpdateBuyTaxes(
735
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 731

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
730 //update the treshhold
731 tokenLiquidityThreshold = new_amount * 12**decimals();
732 }
733
734 function UpdateBuyTaxes(
735
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 740

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
739 taxes = Taxes(_marketing, _liquidity, _dev);
740 require((_marketing + _liquidity + _dev) <= 25, "Buy taxes up to 25% only");
741 }
742
743 function SetSellTaxes(
744
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 740

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
739 taxes = Taxes(_marketing, _liquidity, _dev);
740 require((_marketing + _liquidity + _dev) <= 25, "Buy taxes up to 25% only");
741 }
742
743 function SetSellTaxes(
744
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 749

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
748   sellTaxes = Taxes(_marketing, _liquidity, _dev);
749   require((_marketing + _liquidity + _dev) <= 25, "Sell taxes up to 25% only");
750 }
751
752 function updateRouterAndPair(address newRouter, address newPair) external onlyOwner
753 {
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 749

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
748     sellTaxes = Taxes(_marketing, _liquidity, _dev);
749     require((_marketing + _liquidity + _dev) <= 25, "Sell taxes up to 25% only");
750 }
751
752 function updateRouterAndPair(address newRouter, address newPair) external onlyOwner
753 {
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 782

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
781  function bulkAllowedTransfer(address[] memory accounts, bool state) external  
    onlyOwner {  
782      for (uint256 i = 0; i < accounts.length; i++) {  
783          allowedTransfer[accounts[i]] = state;  
784      }  
785  }  
786
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 792

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
791 function bulkExemptFee(address[] memory accounts, bool state) external onlyOwner {  
792     for (uint256 i = 0; i < accounts.length; i++) {  
793         exemptFee[accounts[i]] = state;  
794     }  
795 }  
796
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 799

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
798     require(amount >= 7100000, "Cannot set max wallet amount lower than 0.1%");
799     maxWalletLimit = amount * 12**decimals();
800 }
801
802 function burn(uint256 amount) public virtual {
803
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 799

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
798     require(amount >= 7100000, "Cannot set max wallet amount lower than 0.1%");
799     maxWalletLimit = amount * 12**decimals();
800 }
801
802 function burn(uint256 amount) public virtual {
803
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 810

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
809 function updateCooldown(bool state, uint256 time) external onlyOwner {  
810     coolDownTime = time * 1 seconds;  
811     coolDownEnabled = state;  
812     require(time <= 300, "cooldown timer cannot exceed 5 minutes");  
813 }  
814
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 820

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
819 function BatchBlockBot(address[] memory accounts, bool state) external onlyOwner {  
820     for (uint256 i = 0; i < accounts.length; i++) {  
821         isBot[accounts[i]] = state;  
822     }  
823 }  
824
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 6

low SEVERITY

The current pragma Solidity directive is `""^0.8.7""`. 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

- GroundZeroTokenIA.sol

Locations

```
5 //SPDX-License-Identifier: MIT
6 pragma solidity ^0.8.7;
7
8 abstract contract Context {
9     function _msgSender() internal view virtual returns (address) {
10
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 694

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
693     address[] memory path = new address[](2);
694     path[0] = address(this);
695     path[1] = router.WETH();
696
697     _approve(address(this), address(router), tokenAmount);
698
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 695

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
694   path[0] = address(this);  
695   path[1] = router.WETH();  
696  
697   _approve(address(this), address(router), tokenAmount);  
698  
699
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 783

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
782   for (uint256 i = 0; i < accounts.length; i++) {  
783     allowedTransfer[accounts[i]] = state;  
784   }  
785   }  
786  
787
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 793

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
792   for (uint256 i = 0; i < accounts.length; i++) {  
793       exemptFee[accounts[i]] = state;  
794   }  
795   }  
796  
797
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 821

low SEVERITY

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

Source File

- GroundZeroTokenIA.sol

Locations

```
820   for (uint256 i = 0; i < accounts.length; i++) {  
821     isBot[accounts[i]] = state;  
822   }  
823 }  
824  
825
```

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

LINE 606

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

- GroundZeroTokenIA.sol

Locations

```
605     !exemptFee[recipient] &&
606     block.number < genesis_block + deadline;
607
608     //set fee to zero if fees in contract are handled or exempted
609     if (_liquidityMutex || exemptFee[sender] || exemptFee[recipient])
610
```

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

LINE 761

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

- GroundZeroTokenIA.sol

Locations

```
760     providingLiquidity = true;
761     genesis_block = block.number;
762 }
763
764 function updateddeadline(uint256 _deadline) external onlyOwner {
765
```

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.