SuperBowl GG

Smart Contract Audit Report

SYSFIXED

11 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
SuperBowl GG	SUPER	Binance Smart Chain

Addresses

Contract address	0xB67A4E3687536fD6CD9DCba897Db6AfC98204528
Contract deployer address	0x41CdB931a0eD0D3B9276de1fCAacB16b1DFd7F90

Project Website

https://superbowl.gg/

Codebase

https://bscscan.com/address/0xB67A4E3687536fD6CD9DCba897Db6AfC98204528#code

SUMMARY

Missed your chance to be a part of the World Cup through crypto? Introducing Superbowl GG A Superbowl orientated token with 2 Staking pools with high APYs BUSD Reward Pool for the winning staking pool!

Contract Summary

Documentation Quality

SuperBowl GG 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 SuperBowl GG 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 105, 170 and 182.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 119, 119, 177, 177, 178, 178, 306, 334, 374, 374, 396, 406, 406, 407, 419, 419, 419, 419, 420, 420, 424, 424, 424, 425, 425, 429, 429, 433, 433, 437, 437, 441, 441, 442, 442, 444, 444, 445, 446, 522, 536, 536, 572, 572, 573, 573, 574, 574, 609, 609, 610, 610, 627, 628, 628, 629, 629, 643, 645, 669, 669, 671 and 675.
- 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 557, 558, 628, 629 and 629.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 483.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 606.

CONCLUSION

We have audited the SuperBowl GG project released on January 2023 to discover issues and identify potential security vulnerabilities in SuperBowl GG 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 SuperBowl GG 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
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
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	Delegate calls 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
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
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

SMART CONTRACT ANALYSIS

Started	Tuesday Jan 10 2023 09:10:50 GMT+0000 (Coordinated Universal Time)
Finished	Wednesday Jan 11 2023 02:07:32 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	SuperBowlGG.sol

Detected Issues

ID	Title	Severity	Status
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-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-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-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 119

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
118   uint8 constant private _decimals = 18;
119   uint256 constant private _tTotal = startingSupply * 10**_decimals;
120
121   struct Fees {
122   uint16 buyFee;
123
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 119

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
118   uint8 constant private _decimals = 18;
119   uint256 constant private _tTotal = startingSupply * 10**_decimals;
120
121   struct Fees {
122   uint16 buyFee;
123
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 177

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
176
177   uint256 private _maxTxAmount = (_tTotal * 100) / 100;
178   uint256 private _maxWalletSize = (_tTotal * 100) / 100;
179
180   bool public tradingEnabled = false;
181
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 177

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
176
177   uint256 private _maxTxAmount = (_tTotal * 100) / 100;
178   uint256 private _maxWalletSize = (_tTotal * 100) / 100;
179
180   bool public tradingEnabled = false;
181
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 178

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
177   uint256 private _maxTxAmount = (_tTotal * 100) / 100;
178   uint256 private _maxWalletSize = (_tTotal * 100) / 100;
179
180   bool public tradingEnabled = false;
181   bool public _hasLiqBeenAdded = false;
182
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 178

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
177   uint256 private _maxTxAmount = (_tTotal * 100) / 100;
178   uint256 private _maxWalletSize = (_tTotal * 100) / 100;
179
180   bool public tradingEnabled = false;
181   bool public _hasLiqBeenAdded = false;
182
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 306

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
305   if (_allowances[sender][msg.sender] != type(uint256).max) {
306   _allowances[sender][msg.sender] -= amount;
307   }
308
309   return _transfer(sender, recipient, amount);
310
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 334

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
333   if (timeSinceLastPair != 0) {
334   require(block.timestamp - timeSinceLastPair > 3 days, "3 Day cooldown.");
335   }
336   require(!lpPairs[pair], "Pair already added to list.");
337   lpPairs[pair] = true;
338
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 374

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
373   function getCirculatingSupply() public view returns (uint256) {
374   return (_tTotal - (balanceOf(DEAD) + balanceOf(address(0))));
375   }
376
377   function removeSniper(address account) external onlyOwner {
378
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 374

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
373   function getCirculatingSupply() public view returns (uint256) {
374   return (_tTotal - (balanceOf(DEAD) + balanceOf(address(0))));
375   }
376
377   function removeSniper(address account) external onlyOwner {
378
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 396

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
395   "Cannot exceed maximums.");
396   require(buyFee + sellFee <= maxRoundtripTax, "Cannot exceed roundtrip maximum.");
397   _taxRates.buyFee = buyFee;
398   _taxRates.sellFee = sellFee;
399   _taxRates.transferFee = transferFee;
400
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 406

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
405   _ratios.staking = staking;
406   _ratios.totalSwap = staking + marketing + staking;
407   uint256 total = _taxRates.buyFee + _taxRates.sellFee;
408   require(_ratios.totalSwap <= total, "Cannot exceed sum of buy and sell fees.");
409   }
410
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 406

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
405   _ratios.staking = staking;
406   _ratios.totalSwap = staking + marketing + staking;
407   uint256 total = _taxRates.buyFee + _taxRates.sellFee;
408   require(_ratios.totalSwap <= total, "Cannot exceed sum of buy and sell fees.");
409   }
410
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 407

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
406   _ratios.totalSwap = staking + marketing + staking;
407   uint256 total = _taxRates.buyFee + _taxRates.sellFee;
408   require(_ratios.totalSwap <= total, "Cannot exceed sum of buy and sell fees.");
409   }
410
411
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 419

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
418   function setMaxTxPercent(uint256 percent, uint256 divisor) external onlyOwner {
419   require((_tTotal * percent) / divisor >= (_tTotal * 5 / 1000), "Max Transaction amt
must be above 0.5% of total supply.");
420   _maxTxAmount = (_tTotal * percent) / divisor;
421   }
422
423
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 419

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
418   function setMaxTxPercent(uint256 percent, uint256 divisor) external onlyOwner {
419   require((_tTotal * percent) / divisor >= (_tTotal * 5 / 1000), "Max Transaction amt
must be above 0.5% of total supply.");
420   _maxTxAmount = (_tTotal * percent) / divisor;
421   }
422
423
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 419

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
418   function setMaxTxPercent(uint256 percent, uint256 divisor) external onlyOwner {
419   require((_tTotal * percent) / divisor >= (_tTotal * 5 / 1000), "Max Transaction amt
must be above 0.5% of total supply.");
420   _maxTxAmount = (_tTotal * percent) / divisor;
421   }
422
423
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 419

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
418   function setMaxTxPercent(uint256 percent, uint256 divisor) external onlyOwner {
419   require((_tTotal * percent) / divisor >= (_tTotal * 5 / 1000), "Max Transaction amt
must be above 0.5% of total supply.");
420   _maxTxAmount = (_tTotal * percent) / divisor;
421   }
422
423
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 420

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
419   require((_tTotal * percent) / divisor >= (_tTotal * 5 / 1000), "Max Transaction amt
must be above 0.5% of total supply.");
420   _maxTxAmount = (_tTotal * percent) / divisor;
421   }
422
423   function setMaxWalletSize(uint256 percent, uint256 divisor) external onlyOwner {
424
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 420

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
419   require((_tTotal * percent) / divisor >= (_tTotal * 5 / 1000), "Max Transaction amt
must be above 0.5% of total supply.");
420   _maxTxAmount = (_tTotal * percent) / divisor;
421   }
422
423   function setMaxWalletSize(uint256 percent, uint256 divisor) external onlyOwner {
424
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 424

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
423   function setMaxWalletSize(uint256 percent, uint256 divisor) external onlyOwner {
424   require((_tTotal * percent) / divisor >= (_tTotal / 100), "Max Wallet amt must be
above 1% of total supply.");
425   _maxWalletSize = (_tTotal * percent) / divisor;
426   }
427
428
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 424

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
423   function setMaxWalletSize(uint256 percent, uint256 divisor) external onlyOwner {
424   require((_tTotal * percent) / divisor >= (_tTotal / 100), "Max Wallet amt must be
above 1% of total supply.");
425   _maxWalletSize = (_tTotal * percent) / divisor;
426   }
427
428
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 424

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
423   function setMaxWalletSize(uint256 percent, uint256 divisor) external onlyOwner {
424   require((_tTotal * percent) / divisor >= (_tTotal / 100), "Max Wallet amt must be
above 1% of total supply.");
425   _maxWalletSize = (_tTotal * percent) / divisor;
426   }
427
428
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 425

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
424   require((_tTotal * percent) / divisor >= (_tTotal / 100), "Max Wallet amt must be
above 1% of total supply.");
425   _maxWalletSize = (_tTotal * percent) / divisor;
426   }
427
428   function getMaxTX() external view returns (uint256) {
429
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 425

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
424   require((_tTotal * percent) / divisor >= (_tTotal / 100), "Max Wallet amt must be
above 1% of total supply.");
425   _maxWalletSize = (_tTotal * percent) / divisor;
426   }
427
428   function getMaxTX() external view returns (uint256) {
429
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 429

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
428   function getMaxTX() external view returns (uint256) {
429   return _maxTxAmount / (10**_decimals);
430   }
431
432   function getMaxWallet() external view returns (uint256) {
433
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 429

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
428   function getMaxTX() external view returns (uint256) {
429   return _maxTxAmount / (10**_decimals);
430   }
431
432   function getMaxWallet() external view returns (uint256) {
433
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 433

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
432   function getMaxWallet() external view returns (uint256) {
433   return _maxWalletSize / (10**_decimals);
434   }
435
436   function getTokenAmountAtPriceImpact(uint256 priceImpactInHundreds) external view
returns (uint256) {
437
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 433

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
432   function getMaxWallet() external view returns (uint256) {
433   return _maxWalletSize / (10**_decimals);
434   }
435
436   function getTokenAmountAtPriceImpact(uint256 priceImpactInHundreds) external view
returns (uint256) {
437
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 437

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
436   function getTokenAmountAtPriceImpact(uint256 priceImpactInHundreds) external view
returns (uint256) {
437   return((balanceOf(lpPair) * priceImpactInHundreds) / masterTaxDivisor);
438   }
439
440   function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
441
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 437

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
436   function getTokenAmountAtPriceImpact(uint256 priceImpactInHundreds) external view
returns (uint256) {
437   return((balanceOf(lpPair) * priceImpactInHundreds) / masterTaxDivisor);
438   }
439
440   function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
441
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 441

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
440   function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
441   swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
442   swapAmount = (_tTotal * amountPercent) / amountDivisor;
443   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
444   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
445
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 441

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
440   function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
441   swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
442   swapAmount = (_tTotal * amountPercent) / amountDivisor;
443   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
444   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
445
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 442

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
441   swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
442   swapAmount = (_tTotal * amountPercent) / amountDivisor;
443   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
444   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
445   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
446
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 442

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
441   swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
442   swapAmount = (_tTotal * amountPercent) / amountDivisor;
443   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
444   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
445   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
446
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 444

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
443   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
444   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
445   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
446   require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
447   }
448
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 444

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
443   require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
444   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
445   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
446   require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
447   }
448
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 445

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
444   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
445   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
446   require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
447   }
448
449
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 446

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
445   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
446   require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
447   }
448
449   function setPriceImpactSwapAmount(uint256 priceImpactSwapPercent) external
onlyOwner {
450
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 522

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
521   if (!_isExcludedFromLimits[to]) {
522   require(balanceOf(to) + amount <= _maxWalletSize, "Transfer amount exceeds the
maxWalletSize.");
523   }
524   }
525   }
526
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 536

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
535   uint256 swapAmt = swapAmount;
536   if (piContractSwapsEnabled) { swapAmt = (balanceOf(lpPair) * piSwapPercent) /
masterTaxDivisor; }
537   if (contractTokenBalance >= swapAmt) { contractTokenBalance = swapAmt; }
538   contractSwap(contractTokenBalance);
539   }
540
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 536

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
535   uint256 swapAmt = swapAmount;
536   if (piContractSwapsEnabled) { swapAmt = (balanceOf(lpPair) * piSwapPercent) /
masterTaxDivisor; }
537   if (contractTokenBalance >= swapAmt) { contractTokenBalance = swapAmt; }
538   contractSwap(contractTokenBalance);
539   }
540
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 572

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
571   bool success;
572   uint256 stakingBalance = (amtBalance * ratios.staking) / ratios.totalSwap;
573   uint256 teamBalance = (amtBalance * ratios.team) / ratios.totalSwap;
574   uint256 marketingBalance = amtBalance - (stakingBalance + teamBalance);
575   if (ratios.marketing > 0) {
576
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 572

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
571   bool success;
572   uint256 stakingBalance = (amtBalance * ratios.staking) / ratios.totalSwap;
573   uint256 teamBalance = (amtBalance * ratios.team) / ratios.totalSwap;
574   uint256 marketingBalance = amtBalance - (stakingBalance + teamBalance);
575   if (ratios.marketing > 0) {
576
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 573

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
572   uint256 stakingBalance = (amtBalance * ratios.staking) / ratios.totalSwap;
573   uint256 teamBalance = (amtBalance * ratios.team) / ratios.totalSwap;
574   uint256 marketingBalance = amtBalance - (stakingBalance + teamBalance);
575   if (ratios.marketing > 0) {
576   (success,) = _taxWallets.marketing.call{value: marketingBalance, gas: 55000}("");
577
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 573

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
572   uint256 stakingBalance = (amtBalance * ratios.staking) / ratios.totalSwap;
573   uint256 teamBalance = (amtBalance * ratios.team) / ratios.totalSwap;
574   uint256 marketingBalance = amtBalance - (stakingBalance + teamBalance);
575   if (ratios.marketing > 0) {
576   (success,) = _taxWallets.marketing.call{value: marketingBalance, gas: 55000}("");
577
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 574

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
573   uint256 teamBalance = (amtBalance * ratios.team) / ratios.totalSwap;
574   uint256 marketingBalance = amtBalance - (stakingBalance + teamBalance);
575   if (ratios.marketing > 0) {
576   (success,) = _taxWallets.marketing.call{value: marketingBalance, gas: 55000}("");
577   }
578
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 574

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
573   uint256 teamBalance = (amtBalance * ratios.team) / ratios.totalSwap;
574   uint256 marketingBalance = amtBalance - (stakingBalance + teamBalance);
575   if (ratios.marketing > 0) {
576   (success,) = _taxWallets.marketing.call{value: marketingBalance, gas: 55000}("");
577   }
578
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 609

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
608   allowedPresaleExclusion = false;
609   swapThreshold = (balanceOf(lpPair) * 10) / 10000;
610   swapAmount = (balanceOf(lpPair) * 30) / 10000;
611   launchStamp = block.timestamp;
612   }
613
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 609

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
608   allowedPresaleExclusion = false;
609   swapThreshold = (balanceOf(lpPair) * 10) / 10000;
610   swapAmount = (balanceOf(lpPair) * 30) / 10000;
611   launchStamp = block.timestamp;
612   }
613
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 610

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
609   swapThreshold = (balanceOf(lpPair) * 10) / 10000;
610   swapAmount = (balanceOf(lpPair) * 30) / 10000;
611   launchStamp = block.timestamp;
612   }
613
614
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 610

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
609   swapThreshold = (balanceOf(lpPair) * 10) / 10000;
610   swapAmount = (balanceOf(lpPair) * 30) / 10000;
611   launchStamp = block.timestamp;
612   }
613
614
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 627

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
626   require(accounts.length == amounts.length, "Lengths do not match.");
627   for (uint16 i = 0; i < accounts.length; i++) {
628   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
629   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
630   }
631
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 628

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
627   for (uint16 i = 0; i < accounts.length; i++) {
628   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
629   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
630   }
631   }
632
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 628

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
627   for (uint16 i = 0; i < accounts.length; i++) {
628   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
629   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
630   }
631   }
632
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 629

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
628   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
629   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
630   }
631   }
632
633
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 629

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
628   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
629   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
630   }
631   }
632
633
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 643

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
642   }
643   _tOwned[from] -= amount;
644   uint256 amountReceived = (takeFee) ? takeTaxes(from, buy, sell, amount) : amount;
645   _tOwned[to] += amountReceived;
646   emit Transfer(from, to, amountReceived);
647
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 645

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
644   uint256 amountReceived = (takeFee) ? takeTaxes(from, buy, sell, amount) : amount;
645   _tOwned[to] += amountReceived;
646   emit Transfer(from, to, amountReceived);
647   if (!_hasLiqBeenAdded) {
648   _checkLiquidityAdd(from, to);
649
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 669

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
668   || block.chainid == 56)) { currentFee = 4500; }
669   uint256 feeAmount = amount * currentFee / masterTaxDivisor;
670   if (feeAmount > 0) {
671   _tOwned[address(this)] += feeAmount;
672   emit Transfer(from, address(this), feeAmount);
673
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 669

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
668   || block.chainid == 56)) { currentFee = 4500; }
669   uint256 feeAmount = amount * currentFee / masterTaxDivisor;
670   if (feeAmount > 0) {
671   _tOwned[address(this)] += feeAmount;
672   emit Transfer(from, address(this), feeAmount);
673
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 671

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
670   if (feeAmount > 0) {
671   _tOwned[address(this)] += feeAmount;
672   emit Transfer(from, address(this), feeAmount);
673   }
674
675
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 675

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
674
675   return amount - feeAmount;
676   }
677   }
678
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 6

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

- SuperBowlGG.sol

Locations

```
5   // SPDX-License-Identifier: MIT
6   pragma solidity >=0.6.0 <0.9.0;
7
8   interface IERC20 {
9   function totalSupply() external view returns (uint256);
10
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 105

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

- SuperBowlGG.sol

Locations

```
104   mapping (address => uint256) private _tOwned;
105   mapping (address => bool) lpPairs;
106   uint256 private timeSinceLastPair = 0;
107   mapping (address => mapping (address => uint256)) private _allowances;
108   mapping (address => bool) private _liquidityHolders;
109
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 170

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

- SuperBowlGG.sol

Locations

```
169
170   bool inSwap;
171   bool public contractSwapEnabled = false;
172   uint256 public swapThreshold;
173   uint256 public swapAmount;
174
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 182

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
181   bool public _hasLiqBeenAdded = false;
182   Protections protections;
183   uint256 public launchStamp;
184
185   event ContractSwapEnabledUpdated(bool enabled);
186
```

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

LINE 483

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

- SuperBowlGG.sol

Locations

```
482   && to != _owner
483   && tx.origin != _owner
484   && !_liquidityHolders[to]
485   && !_liquidityHolders[from]
486   && to != DEAD
487
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 557

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
556   address[] memory path = new address[](2);
557   path[0] = address(this);
558   path[1] = dexRouter.WETH();
559
560   try dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(
561
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 558

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
557   path[0] = address(this);
558   path[1] = dexRouter.WETH();
559
560   try dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(
561   contractTokenBalance,
562
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 628

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
627   for (uint16 i = 0; i < accounts.length; i++) {
628   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
629   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
630   }
631   }
632
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 629

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
628   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
629   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
630   }
631   }
632
633
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 629

low SEVERITY

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

Source File

- SuperBowlGG.sol

Locations

```
628   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
629   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
630   }
631   }
632
633
```

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

- SuperBowlGG.sol

Locations

```
605   }
606   try protections.setLaunch(lpPair, uint32(block.number), uint64(block.timestamp),
_decimals) {} catch {}
607   tradingEnabled = true;
608   allowedPresaleExclusion = false;
609   swapThreshold = (balanceOf(lpPair) * 10) / 10000;
610
```

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.