Howl Finance

Smart Contract Audit Report

SYSFIXED

22 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Howl Finance	HOWL	BSC

Addresses

Contract address	0xD0343DF21e5019769456f211f5baF3cFc0A4597C
Contract deployer address	0xEEEedBa6Bfbd03eFE1De85988f9db3a1edeafa52

Project Website

https://www.howlfinance.com/

Codebase

https://bscscan.com/address/0xD0343DF21e5019769456f211f5baF3cFc0A4597C#contracts

SUMMARY

Howl Finance is a utility token with 7 live usecases. including Liquidity Farming, Staking pools, Swap, Portfolio for NFT and Tokens Value, NFT BUSD rewards, NFT Rarity and Software as a Service.

Contract Summary

Documentation Quality

This project has a standard of documentation.

- Technical description provided.

Code Quality

The quality of the code in this project is up to standard.

- The official Solidity style guide is followed.

Test Scope

Project test coverage is 100% (Via Codebase).

Audit Findings Summary

Issues Found

- SWC-101 | Arithmetic operation issues discovered on lines 21, 31, 40, 41, 50, 199, 201, 202, 215, 236, 300, 305, 317, 357, 383, 413, 415, 416, 432, 434, 435, 436, 456, 464, 472, 481, 503, 519, 520, and 529.
- SWC-108 | State variable visibility is not set on lines 191, 205, 218, 219, 220, and 237. It is best practice to set the visibility of state variables explicitly to public or private.
- SWC-110 | Out of bounds array access issues discovered on lines 419, 420, 457, 458, 465, 466, 473, and 474.

CONCLUSION

We have audited the Howl Finance project which has released on January 2023 to discover issues and identify potential security vulnerabilities in Howl Finance Project. This process is used to find technical issues and security loopholes that find some common issues in the code.

The security audit report produced satisfactory results with low-risk issues.

The most common issue found in writing code on contracts that do not pose a big risk is that writing on contracts is close to the standard of writing contracts in general. The low-level issue found is a state variable visibility is not set, and out of bounds array access which the index access expression can cause an exception in case of 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.	PASS
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
Check-Effect Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	PASS
Assert Violation	SWC-110	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 Caller	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
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	PASS
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	Sat Jan 21 2023 23:12:43 GMT+0000 (Coordinated Universal Time)
Finished	Sun Jan 22 2023 01:32:12 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	HowlFinance.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-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-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT 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-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-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 21

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
20   function add(uint256 a, uint256 b) internal pure returns (uint256) {
21   uint256 c = a + b;
22   require(c >= a, "SafeMath: addition overflow");
23
24   return c;
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 31

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
30   require(b <= a, errorMessage);
31   uint256 c = a - b;
32
33   return c;
34   }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 40

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
39
40   uint256 c = a * b;
41   require(c / a == b, "SafeMath: multiplication overflow");
42
43   return c;
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 41

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
40   uint256 c = a * b;
41   require(c / a == b, "SafeMath: multiplication overflow");
42
43   return c;
44   }
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 50

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
49   require(b > 0, errorMessage);
50   uint256 c = a / b;
51   return c;
52   }
53   }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 199

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
198
199   uint256 public _totalSupply = 1 * 10**9 * 10**_decimals;
200
201   uint256 public _maxTxAmount = _totalSupply / 100; // 1%
202   uint256 public _maxWalletToken = _totalSupply / 50; // 2%
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 201

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
200
201   uint256 public _maxTxAmount = _totalSupply / 100; // 1%
202   uint256 public _maxWalletToken = _totalSupply / 50; // 2%
203
204   mapping (address => uint256) public balanceOf;
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 202

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
201   uint256 public _maxTxAmount = _totalSupply / 100; // 1%
202   uint256 public _maxWalletToken = _totalSupply / 50; // 2%
203
204   mapping (address => uint256) public balanceOf;
205   mapping (address => mapping (address => uint256)) _allowances;
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 215

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
214   uint256 public burnFee = 0;
215   uint256 public totalFee = marketingFee + liquidityFee + buybackFee + burnFee;
216   uint256 public constant feeDenominator = 1000;
217
218   uint256 sellMultiplier = 100;
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 236

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
235   bool public swapEnabled = true;
236   uint256 public swapThreshold = _totalSupply / 1000;
237   bool inSwap;
238   modifier swapping() { inSwap = true; _; inSwap = false; }
239
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 300

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
299   require(maxWallPercent_base10000 >= 10,"Cannot set max wallet less than 0.1%");
300   _maxWalletToken = (_totalSupply * maxWallPercent_base10000 ) / 10000;
301   emit config_MaxWallet(_maxWalletToken);
302   }
303   function setMaxTxPercent_base10000(uint256 maxTXPercentage_base10000) external
onlyOwner {
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 305

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
304   require(maxTXPercentage_base10000 >= 10,"Cannot set max transaction less than
0.1%");
305   _maxTxAmount = (_totalSupply * maxTXPercentage_base10000 ) / 10000;
306   emit config_MaxTransaction(_maxTxAmount);
307   }
308
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 317

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
316   if (!authorizations[sender] && !isWalletLimitExempt[sender] &&
!isWalletLimitExempt[recipient] && recipient != pair) {
317   require((balanceOf[recipient] + amount) <= _maxWalletToken,"max wallet limit
reached");
318   }
319
320   // Checks max transaction limit
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 356

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
355
356   uint256 feeAmount = amount.mul(totalFee).mul(multiplier).div(feeDenominator * 100);
357   uint256 burnTokens = feeAmount.mul(burnFee).div(totalFee);
358   uint256 contractTokens = feeAmount.sub(burnTokens);
359
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 383

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
382   if(tokenAddress == pair){
383   require(block.timestamp > launchedAt + 500 days,"Locked for 1 year");
384   }
385
386   if(tokens == 0){
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 413

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
412
413   uint256 totalETHFee = totalFee - burnFee;
414
415   uint256 amountToLiquify = (swapThreshold * liquidityFee)/(totalETHFee * 2);
416   uint256 amountToSwap = swapThreshold - amountToLiquify;
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 415

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
414
415   uint256 amountToLiquify = (swapThreshold * liquidityFee)/(totalETHFee * 2);
416   uint256 amountToSwap = swapThreshold - amountToLiquify;
417
418   address[] memory path = new address[](2);
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 416

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
415   uint256 amountToLiquify = (swapThreshold * liquidityFee)/(totalETHFee * 2);
416   uint256 amountToSwap = swapThreshold - amountToLiquify;
417
418   address[] memory path = new address[](2);
419   path[0] = address(this);
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 432

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
431
432   totalETHFee = totalETHFee - (liquidityFee / 2);
433
434   uint256 amountBNBLiquidity = (amountBNB * liquidityFee) / (totalETHFee * 2);
435   uint256 amountBNBMarketing = (amountBNB * marketingFee) / totalETHFee;
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 434

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
433
434   uint256 amountBNBLiquidity = (amountBNB * liquidityFee) / (totalETHFee * 2);
435   uint256 amountBNBMarketing = (amountBNB * marketingFee) / totalETHFee;
436   uint256 amountBNBbuyback = (amountBNB * buybackFee) / totalETHFee;
437
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 435

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
434   uint256 amountBNBLiquidity = (amountBNB * liquidityFee) / (totalETHFee * 2);
435   uint256 amountBNBMarketing = (amountBNB * marketingFee) / totalETHFee;
436   uint256 amountBNBbuyback = (amountBNB * buybackFee) / totalETHFee;
437
438   payable(marketingFeeReceiver).transfer(amountBNBMarketing);
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 436

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
435   uint256 amountBNBMarketing = (amountBNB * marketingFee) / totalETHFee;
436   uint256 amountBNBbuyback = (amountBNB * buybackFee) / totalETHFee;
437
438   payable(marketingFeeReceiver).transfer(amountBNBMarketing);
439   payable(buybackFeeReceiver).transfer(amountBNBbuyback);
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 456

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
455   require(addresses.length < 501,"GAS Error: max limit is 500 addresses");
456   for (uint256 i=0; i < addresses.length; ++i) {
457   isFeeExempt[addresses[i]] = status;
458   emit Wallet_feeExempt(addresses[i], status);
459   }
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 464

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
463   require(addresses.length < 501,"GAS Error: max limit is 500 addresses");
464   for (uint256 i=0; i < addresses.length; ++i) {
465   isTxLimitExempt[addresses[i]] = status;
466   emit Wallet_txExempt(addresses[i], status);
467   }
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 472

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
471   require(addresses.length < 501,"GAS Error: max limit is 500 addresses");
472   for (uint256 i=0; i < addresses.length; ++i) {
473   isWalletLimitExempt[addresses[i]] = status;
474   emit Wallet_holdingExempt(addresses[i], status);
475   }
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 481

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
480   require(totalFee.mul(sellMultiplier).div(100) <= 150, "Sell tax cannot be more than
15%");
481   require(totalFee.mul(sellMultiplier + buyMultiplier).div(100) <= 200, "Buy+Sell tax
cannot be more than 20%");
482   require(totalFee.mul(transferMultiplier).div(100) <= 100, "Transfer Tax cannot be
more than 10%");
483
484   emit UpdateFee( uint8(totalFee.mul(buyMultiplier).div(100)),
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 503

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
502   burnFee = _burnFee;
503   totalFee = _liquidityFee + _marketingFee + _buybackFee + _burnFee;
504
505   update_fees();
506   }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 519

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
518   function setSwapBackSettings(bool _enabled, uint256 _amount) external onlyOwner {
519   require(_amount >= 1 * 10**_decimals, "Amount is less than one token");
520   require(_amount < (_totalSupply/10), "Amount too high");
521
522   swapEnabled = _enabled;
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 520

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
519   require(_amount >= 1 * 10**_decimals, "Amount is less than one token");
520   require(_amount < (_totalSupply/10), "Amount too high");
521
522   swapEnabled = _enabled;
523   swapThreshold = _amount;
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 529

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
528   function getCirculatingSupply() public view returns (uint256) {
529   return (_totalSupply - balanceOf[DEAD] - balanceOf[ZERO]);
530   }
531   /*
532   function LPBurn(uint256 percent_base10000) public authorized returns (bool){
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 191

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
190
191   address immutable WBNB;
192   address constant DEAD = 0x000000000000000000000000000000000000dEaD;
193   address constant ZERO = 0x0000000000000000000000000000000000000000;
194
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 205

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
204   mapping (address => uint256) public balanceOf;
205   mapping (address => mapping (address => uint256)) _allowances;
206
207   mapping (address => bool) public isFeeExempt;
208   mapping (address => bool) public isTxLimitExempt;
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 218

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
217
218   uint256 sellMultiplier = 100;
219   uint256 buyMultiplier = 100;
220   uint256 transferMultiplier = 25;
221
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 219

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
218   uint256 sellMultiplier = 100;
219   uint256 buyMultiplier = 100;
220   uint256 transferMultiplier = 25;
221
222   address public marketingFeeReceiver;
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 220

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
219   uint256 buyMultiplier = 100;
220   uint256 transferMultiplier = 25;
221
222   address public marketingFeeReceiver;
223   address public buybackFeeReceiver;
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 237

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

- HowlFinance.sol

Locations

```
236   uint256 public swapThreshold = _totalSupply / 1000;
237   bool inSwap;
238   modifier swapping() { inSwap = true; _; inSwap = false; }
239
240   constructor () Auth(msg.sender) {
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 419

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
418   address[] memory path = new address[](2);
419   path[0] = address(this);
420   path[1] = WBNB;
421
422   router.swapExactTokensForETHSupportingFeeOnTransferTokens(
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 420

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
419   path[0] = address(this);
420   path[1] = WBNB;
421
422   router.swapExactTokensForETHSupportingFeeOnTransferTokens(
423   amountToSwap,
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 457

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
456   for (uint256 i=0; i < addresses.length; ++i) {
457   isFeeExempt[addresses[i]] = status;
458   emit Wallet_feeExempt(addresses[i], status);
459   }
460   }
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 458

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
457   isFeeExempt[addresses[i]] = status;
458   emit Wallet_feeExempt(addresses[i], status);
459   }
460   }
461
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 465

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
464   for (uint256 i=0; i < addresses.length; ++i) {
465   isTxLimitExempt[addresses[i]] = status;
466   emit Wallet_txExempt(addresses[i], status);
467   }
468   }
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 466

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
465   isTxLimitExempt[addresses[i]] = status;
466   emit Wallet_txExempt(addresses[i], status);
467   }
468   }
469
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 473

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
472   for (uint256 i=0; i < addresses.length; ++i) {
473   isWalletLimitExempt[addresses[i]] = status;
474   emit Wallet_holdingExempt(addresses[i], status);
475   }
476   }
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 474

low SEVERITY

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

Source File

- HowlFinance.sol

Locations

```
473   isWalletLimitExempt[addresses[i]] = status;
474   emit Wallet_holdingExempt(addresses[i], status);
475   }
476   }
477
```

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.