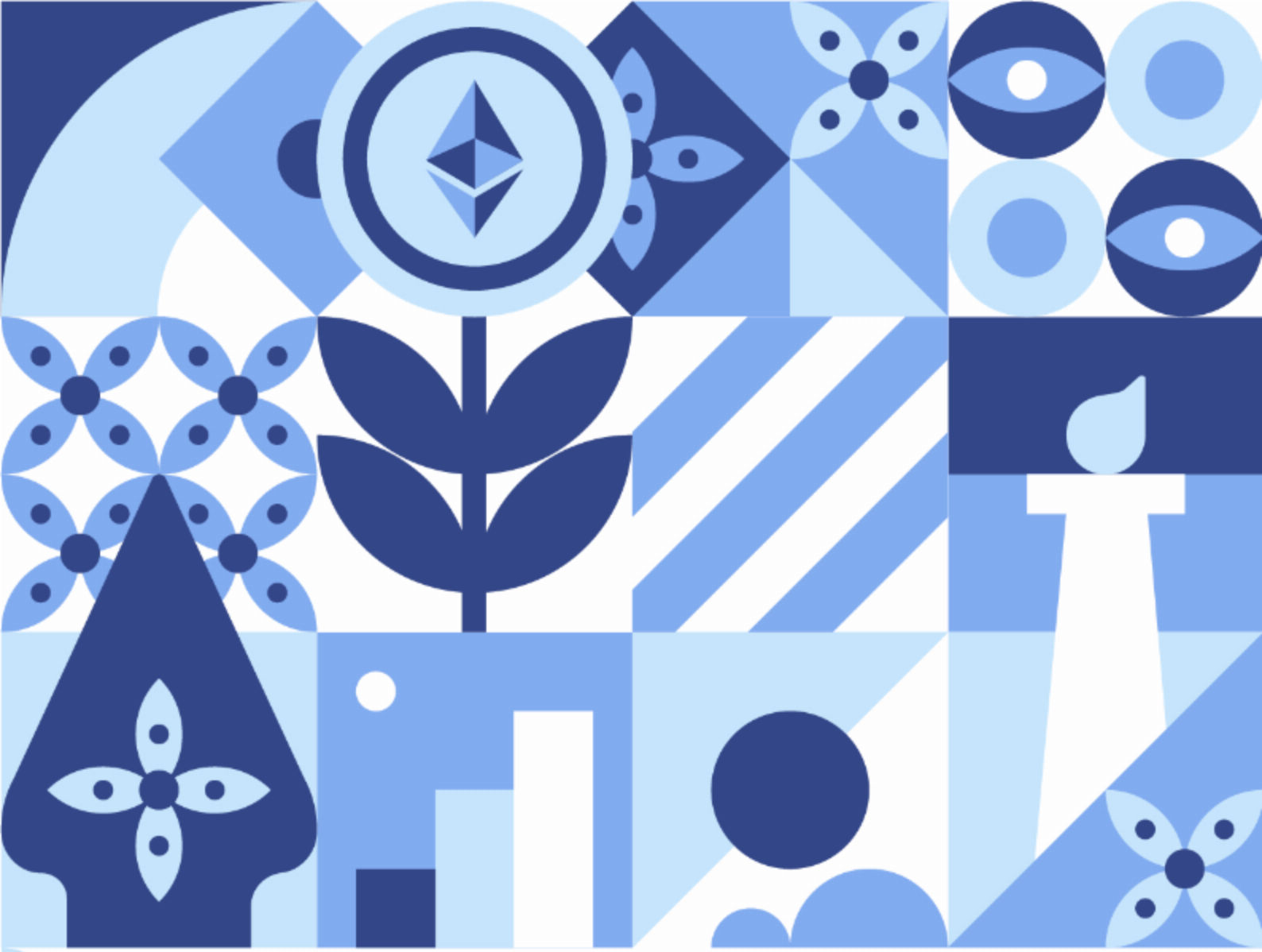

Fable Of The Dragon

Smart Contract Audit Report

SYSFIXED

21 Oct 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Fable Of The Dragon	TYRANT	Ethereum

Addresses

Contract address	0x8ee325ae3e54e83956ef2d5952d3c8bc1fa6ec27
Contract deployer address	0x6BD72A62bd476BC7113010CB939EE39fA80D6a19

Project Website

https://fableofthedragon.com/

Codebase

https://etherscan.io/address/0x8ee325ae3e54e83956ef2d5952d3c8bc1fa6ec27#code

SUMMARY

$TYRANT uses special alchemy to vanquish taxes and fill the King's marketing coffers all at once – leaving the dragon's hoard ready for noble investors to plunder.

Contract Summary

Documentation Quality

Fable Of The Dragon 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 Fable Of The Dragon 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 961.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 769, 793, 809, 812, 834, 836, 887, 906, 909, 949, 949, 950, 950, 959, 959, 960, 960, 985, 985, 1019, 1025, 1038, 1041, 1041, 1042, 1042, 1043, 1043, 1044, 1046, 1056, 1057, 1063, 1082 and 1116.
- 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 1072 and 1073.

CONCLUSION

We have audited the Fable Of The Dragon project released on January 2023 to discover issues and identify potential security vulnerabilities in Fable Of The Dragon 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 Fable Of The Dragon 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 state variable visibility is not set, 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. Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable. Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables.

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
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.	PASS
Write to Arbitrary Storage Location	SWC-124	The contract is responsible for ensuring that only authorized user or contract accounts may write to sensitive storage locations.	PASS
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS
Insufficient Gas Griefing	SWC-126	Insufficient gas griefing attacks can be performed on contracts which accept data and use it in a sub-call on another contract.	PASS
Arbitrary Jump Function	SWC-127	As Solidity doesnt support pointer arithmetics, it is impossible to change such variable to an arbitrary value.	PASS

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

SMART CONTRACT ANALYSIS

Started	Thursday Oct 20 2022 02:56:00 GMT+0000 (Coordinated Universal Time)
Finished	Friday Oct 21 2022 03:53:27 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Tyrant.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-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-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 769

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
768   unchecked {
769   _approve(owner, spender, currentAllowance - subtractedValue);
770   }
771
772   return true;
773
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 793

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
792   address owner = _msgSender();
793   _approve(owner, spender, allowance(owner, spender) + addedValue);
794   return true;
795   }
796
797
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 809

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
808
809   _totalSupply += amount;
810   unchecked {
811   // Overflow not possible: balance + amount is at most totalSupply + amount, which
is checked above.
812   _balances[account] += amount;
813
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 812

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
811   // Overflow not possible: balance + amount is at most totalSupply + amount, which
is checked above.
812   _balances[account] += amount;
813   }
814   emit Transfer(address(0), account, amount);
815   }
816
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 834

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
833   unchecked {
834   _balances[account] = accountBalance - amount;
835   // Overflow not possible: amount <= accountBalance <= totalSupply.
836   _totalSupply -= amount;
837   }
838
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 836

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
835   // Overflow not possible: amount <= accountBalance <= totalSupply.
836   _totalSupply -= amount;
837   }
838
839   emit Transfer(account, address(0), amount);
840
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 887

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
886   unchecked {
887   _approve(owner, spender, currentAllowance - amount);
888   }
889   }
890   }
891
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 906

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
905   unchecked {
906   _balances[from] = fromBalance - amount;
907   // Overflow not possible: the sum of all balances is capped by totalSupply, and the
sum is preserved by
908   // decrementing then incrementing.
909   _balances[to] += amount;
910
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 909

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
908   // decrementing then incrementing.
909   _balances[to] += amount;
910   }
911
912   emit Transfer(from, to, amount);
913
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 949

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
948   uint256 public taxForMarketing = 25;
949   uint256 public maxTxAmount = 100000 * 10**_decimals;
950   uint256 public maxWalletAmount = 100000 * 10**_decimals;
951   address public marketingWallet = 0x6BD72A62bd476BC7113010CB939EE39fA80D6a19;
952   // TOKENOMICS END ============================================================>
953
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 949

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
948   uint256 public taxForMarketing = 25;
949   uint256 public maxTxAmount = 100000 * 10**_decimals;
950   uint256 public maxWalletAmount = 100000 * 10**_decimals;
951   address public marketingWallet = 0x6BD72A62bd476BC7113010CB939EE39fA80D6a19;
952   // TOKENOMICS END ============================================================>
953
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 950

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
949   uint256 public maxTxAmount = 100000 * 10**_decimals;
950   uint256 public maxWalletAmount = 100000 * 10**_decimals;
951   address public marketingWallet = 0x6BD72A62bd476BC7113010CB939EE39fA80D6a19;
952   // TOKENOMICS END ===============================================================>
953
954
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 950

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
949   uint256 public maxTxAmount = 100000 * 10**_decimals;
950   uint256 public maxWalletAmount = 100000 * 10**_decimals;
951   address public marketingWallet = 0x6BD72A62bd476BC7113010CB939EE39fA80D6a19;
952   // TOKENOMICS END ===============================================================>
953
954
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 959

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
958   mapping(address => bool) private _isExcludedFromFee;
959   uint256 private _numTokensSellToAddToLiquidity = 5000 * 10**_decimals;
960   uint256 private _numTokensSellToAddToETH = 2000 * 10**_decimals;
961   bool inSwapAndLiquify;
962
963
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 959

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
958   mapping(address => bool) private _isExcludedFromFee;
959   uint256 private _numTokensSellToAddToLiquidity = 5000 * 10**_decimals;
960   uint256 private _numTokensSellToAddToETH = 2000 * 10**_decimals;
961   bool inSwapAndLiquify;
962
963
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 960

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
959   uint256 private _numTokensSellToAddToLiquidity = 5000 * 10**_decimals;
960   uint256 private _numTokensSellToAddToETH = 2000 * 10**_decimals;
961   bool inSwapAndLiquify;
962
963   event SwapAndLiquify(
964
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 960

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
959   uint256 private _numTokensSellToAddToLiquidity = 5000 * 10**_decimals;
960   uint256 private _numTokensSellToAddToETH = 2000 * 10**_decimals;
961   bool inSwapAndLiquify;
962
963   event SwapAndLiquify(
964
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 985

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
984   constructor() ERC20(_name, _symbol) {
985   _mint(msg.sender, (_supply * 10**_decimals));
986
987   IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D);
988   uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),
_uniswapV2Router.WETH());
989
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 985

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
984   constructor() ERC20(_name, _symbol) {
985   _mint(msg.sender, (_supply * 10**_decimals));
986
987   IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D);
988   uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),
_uniswapV2Router.WETH());
989
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1019

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
1018   if (from != uniswapV2Pair) {
1019   uint256 contractLiquidityBalance = balanceOf(address(this)) - _marketingReserves;
1020   if (contractLiquidityBalance >= _numTokensSellToAddToLiquidity) {
1021   _swapAndLiquify(_numTokensSellToAddToLiquidity);
1022   }
1023
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 1025

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
1024   _swapTokensForEth(_numTokensSellToAddToETH);
1025   _marketingReserves -= _numTokensSellToAddToETH;
1026   bool sent = payable(marketingWallet).send(address(this).balance);
1027   require(sent, "Failed to send ETH");
1028   }
1029
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1038

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
1037   if(from == uniswapV2Pair){
1038   require((amount + balanceOf(to)) <= maxWalletAmount, "ERC20: balance amount
exceeded max wallet amount limit");
1039   }
1040
1041   uint256 marketingShare = ((amount * taxForMarketing) / 100);
1042
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1041

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
1040
1041   uint256 marketingShare = ((amount * taxForMarketing) / 100);
1042   uint256 liquidityShare = ((amount * taxForLiquidity) / 100);
1043   transferAmount = amount - (marketingShare + liquidityShare);
1044   _marketingReserves += marketingShare;
1045
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1041

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
1040
1041   uint256 marketingShare = ((amount * taxForMarketing) / 100);
1042   uint256 liquidityShare = ((amount * taxForLiquidity) / 100);
1043   transferAmount = amount - (marketingShare + liquidityShare);
1044   _marketingReserves += marketingShare;
1045
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1042

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
1041   uint256 marketingShare = ((amount * taxForMarketing) / 100);
1042   uint256 liquidityShare = ((amount * taxForLiquidity) / 100);
1043   transferAmount = amount - (marketingShare + liquidityShare);
1044   _marketingReserves += marketingShare;
1045
1046
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1042

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
1041   uint256 marketingShare = ((amount * taxForMarketing) / 100);
1042   uint256 liquidityShare = ((amount * taxForLiquidity) / 100);
1043   transferAmount = amount - (marketingShare + liquidityShare);
1044   _marketingReserves += marketingShare;
1045
1046
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1043

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
1042   uint256 liquidityShare = ((amount * taxForLiquidity) / 100);
1043   transferAmount = amount - (marketingShare + liquidityShare);
1044   _marketingReserves += marketingShare;
1045
1046   super._transfer(from, address(this), (marketingShare + liquidityShare));
1047
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1043

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
1042   uint256 liquidityShare = ((amount * taxForLiquidity) / 100);
1043   transferAmount = amount - (marketingShare + liquidityShare);
1044   _marketingReserves += marketingShare;
1045
1046   super._transfer(from, address(this), (marketingShare + liquidityShare));
1047
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1044

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
1043   transferAmount = amount - (marketingShare + liquidityShare);
1044   _marketingReserves += marketingShare;
1045
1046   super._transfer(from, address(this), (marketingShare + liquidityShare));
1047   }
1048
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1046

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
1045
1046   super._transfer(from, address(this), (marketingShare + liquidityShare));
1047   }
1048   super._transfer(from, to, transferAmount);
1049   }
1050
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1056

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
1055   function _swapAndLiquify(uint256 contractTokenBalance) private lockTheSwap {
1056   uint256 half = (contractTokenBalance / 2);
1057   uint256 otherHalf = (contractTokenBalance - half);
1058
1059   uint256 initialBalance = address(this).balance;
1060
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1057

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
1056   uint256 half = (contractTokenBalance / 2);
1057   uint256 otherHalf = (contractTokenBalance - half);
1058
1059   uint256 initialBalance = address(this).balance;
1060
1061
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1063

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
1062
1063   uint256 newBalance = (address(this).balance - initialBalance);
1064
1065   _addLiquidity(otherHalf, newBalance);
1066
1067
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1082

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
1081   address(this),
1082   (block.timestamp + 300)
1083   );
1084   }
1085
1086
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1116

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
1115   {
1116   require((_taxForLiquidity+_taxForMarketing) <= 100, "ERC20: total tax must not be
greater than 100");
1117   taxForLiquidity = _taxForLiquidity;
1118   taxForMarketing = _taxForMarketing;
1119
1120
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 961

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
960   uint256 private _numTokensSellToAddToETH = 2000 * 10**_decimals;
961   bool inSwapAndLiquify;
962
963   event SwapAndLiquify(
964   uint256 tokensSwapped,
965
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1072

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
1071   address[] memory path = new address[](2);
1072   path[0] = address(this);
1073   path[1] = uniswapV2Router.WETH();
1074
1075   _approve(address(this), address(uniswapV2Router), tokenAmount);
1076
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1073

low SEVERITY

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

Source File

- Tyrant.sol

Locations

```
1072   path[0] = address(this);
1073   path[1] = uniswapV2Router.WETH();
1074
1075   _approve(address(this), address(uniswapV2Router), tokenAmount);
1076
1077
```

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.