# YetiCoin

# Smart Contract Audit Report

SYSFIXED

# TABLE OF CONTENTS

# AUDITED DETAILS

## Audited Project

| Project name | Token ticker | Blockchain |
|---|---|---|
| YetiCoin | YETIC | Ethereum |

## Addresses

| Contract address | 0xdf96bde075d59e9143b325c75af38e208c986e6f |
|---|---|
| Contract deployer address | 0xE7412E4dE41676aA29E8F1804B128DF92bFf4323 |

## Project Website

| https://yeticoineth.com/ |
|---|

## Codebase

| https://etherscan.io/address/0xdf96bde075d59e9143b325c75af38e208c986e6f#code |
|---|

# SUMMARY

We are pioneering the M2E (Move2Earn) utility. Earn tokens while you get your fit on!!! We aim to get people moving while partnering with Gyms, Athletes, and different sports organizations to help kids have opportunities they wouldn't normally get. Come check out all the spectacular things we are doing and join us today!!!

## ▌Contract Summary

**Documentation Quality**

YetiCoin 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 YetiCoin 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 358.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 34, 66, 89, 90, 125, 161, 337, 337, 337, 337, 338, 338, 361, 361, 361, 361, 362, 362, 362, 362, 544, 550, 557 and 597.
- 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 497, 497, 497, 498, 498, 498, 498, 498, 498, 498, 661 and 662.

# CONCLUSION

We have audited the YetiCoin project released on November 2021 to discover issues and identify potential security vulnerabilities in YetiCoin 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 YetiCoin 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 and out of bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value.

# AUDIT RESULT

| Article | Category | Description | Result |
|---|---|---|---|
| Default Visibility | SWC-100 SWC-108 | Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously. | ISSUE FOUND |
| Integer Overflow and Underflow | SWC-101 | If unchecked math is used, all math operations should be safe from overflows and underflows. | ISSUE FOUND |
| Outdated Compiler Version | SWC-102 | It is recommended to use a recent version of the Solidity compiler. | PASS |
| Floating Pragma | SWC-103 | Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly. | ISSUE FOUND |
| Unchecked Call Return Value | SWC-104 | The return value of a message call should be checked. | PASS |
| Unprotected Ether Withdrawal | SWC-105 | Due to missing or insufficient access controls, malicious parties can withdraw from the contract. | PASS |
| SELFDESTRUCT Instruction | SWC-106 | The contract should not be self-destructible while it has funds belonging to users. | PASS |
| Reentrancy | SWC-107 | Check effect interaction pattern should be followed if the code performs recursive call. | PASS |
| Uninitialized Storage Pointer | SWC-109 | Uninitialized local storage variables can point to unexpected storage locations in the contract. | PASS |
| Assert Violation | SWC-110 SWC-123 | Properly functioning code should never reach a failing assert statement. | ISSUE FOUND |
| Deprecated Solidity Functions | SWC-111 | Deprecated built-in functions should never be used. | PASS |
| Delegate call to Untrusted Callee | SWC-112 | Delegatecalls should only be allowed to trusted addresses. | PASS |

| DoS (Denial of Service) | SWC-113 SWC-128 | Execution of the code should never be blocked by a specific contract state unless required. | PASS |
|---|---|---|---|
| Race Conditions | SWC-114 | Race Conditions and Transactions Order Dependency should not be possible. | PASS |
| Authorization through tx.origin | SWC-115 | tx.origin should not be used for authorization. | 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 | Tuesday Nov 16 2021 05:20:28 GMT+0000 (Coordinated Universal Time) |
|---|---|
| Finished | Wednesday Nov 17 2021 02:00:04 GMT+0000 (Coordinated Universal Time) |
| Mode | Standard |
| Main Source File | YetiCoin.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-103 | A FLOATING PRAGMA IS 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-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 34

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
33    function add(uint256 a, uint256 b) internal pure returns (uint256) {
34    uint256 c = a + b;
35    require(c >= a, "SafeMath: addition overflow");
36
37    return c;
38
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED
LINE 66

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
65   require(b <= a, errorMessage);
66   uint256 c = a - b;
67
68   return c;
69   }
70
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED
LINE 89

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

## Source File
- YetiCoin.sol

## Locations

```
88
89   uint256 c = a * b;
90   require(c / a == b, "SafeMath: multiplication overflow");
91
92   return c;
93
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED
LINE 90

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
89   uint256 c = a * b;
90   require(c / a == b, "SafeMath: multiplication overflow");
91
92   return c;
93   }
94
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED
LINE 125

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
124    require(b > 0, errorMessage);
125    uint256 c = a / b;
126    // assert(a == b * c + a % b); // There is no case in which this doesn't hold
127
128    return c;
129
```

# SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED
## LINE 161

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
160    require(b != 0, errorMessage);
161    return a % b;
162    }
163    }
164
165
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 337

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
336    uint256 private constant MAX = ~uint256(0);
337    uint256 private _tTotal = 1 * 10**15 * 10**9;
338    uint256 private _rTotal = (MAX - (MAX % _tTotal));
339    uint256 private _tFeeTotal;
340
341
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED
LINE 337

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
336    uint256 private constant MAX = ~uint256(0);
337    uint256 private _tTotal = 1 * 10**15 * 10**9;
338    uint256 private _rTotal = (MAX - (MAX % _tTotal));
339    uint256 private _tFeeTotal;
340
341
```

# SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED
LINE 337

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
336    uint256 private constant MAX = ~uint256(0);
337    uint256 private _tTotal = 1 * 10**15 * 10**9;
338    uint256 private _rTotal = (MAX - (MAX % _tTotal));
339    uint256 private _tFeeTotal;
340
341
```

# SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED
LINE 337

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
336    uint256 private constant MAX = ~uint256(0);
337    uint256 private _tTotal = 1 * 10**15 * 10**9;
338    uint256 private _rTotal = (MAX - (MAX % _tTotal));
339    uint256 private _tFeeTotal;
340
341
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED
LINE 338

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
337   uint256 private _tTotal = 1 * 10**15 * 10**9;
338   uint256 private _rTotal = (MAX - (MAX % _tTotal));
339   uint256 private _tFeeTotal;
340
341   string private _name = "YetiCoin";
342
```

# SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED
LINE 338

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
337   uint256 private _tTotal = 1 * 10**15 * 10**9;
338   uint256 private _rTotal = (MAX - (MAX % _tTotal));
339   uint256 private _tFeeTotal;
340
341   string private _name = "YetiCoin";
342
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED
LINE 361

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
360
361    uint256 public _maxWalletHolding = 25 * 10**12 * 10**9;
362    uint256 private numTokensSellToAddToLiquidity = 9 * 10**12 * 10**9;
363
364    event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
365
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED
LINE 361

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
360
361    uint256 public _maxWalletHolding = 25 * 10**12 * 10**9;
362    uint256 private numTokensSellToAddToLiquidity = 9 * 10**12 * 10**9;
363
364    event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
365
```

# SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 361

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
360
361    uint256 public _maxWalletHolding = 25 * 10**12 * 10**9;
362    uint256 private numTokensSellToAddToLiquidity = 9 * 10**12 * 10**9;
363
364    event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
365
```

# SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED
LINE 361

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

## Source File
- YetiCoin.sol

## Locations

```
360
361    uint256 public _maxWalletHolding = 25 * 10**12 * 10**9;
362    uint256 private numTokensSellToAddToLiquidity = 9 * 10**12 * 10**9;
363
364    event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
365
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED
LINE 362

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
361    uint256 public _maxWalletHolding = 25 * 10**12 * 10**9;
362    uint256 private numTokensSellToAddToLiquidity = 9 * 10**12 * 10**9;
363
364    event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
365    event SwapAndLiquifyEnabledUpdated(bool enabled);
366
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED
LINE 362

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

## Source File
- YetiCoin.sol

## Locations

```
361    uint256 public _maxWalletHolding = 25 * 10**12 * 10**9;
362    uint256 private numTokensSellToAddToLiquidity = 9 * 10**12 * 10**9;
363
364    event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
365    event SwapAndLiquifyEnabledUpdated(bool enabled);
366
```

# SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED
LINE 362

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

## Source File
- YetiCoin.sol

## Locations

```
361    uint256 public _maxWalletHolding = 25 * 10**12 * 10**9;
362    uint256 private numTokensSellToAddToLiquidity = 9 * 10**12 * 10**9;
363
364    event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
365    event SwapAndLiquifyEnabledUpdated(bool enabled);
366
```

# SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED
LINE 362

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
361   uint256 public _maxWalletHolding = 25 * 10**12 * 10**9;
362   uint256 private numTokensSellToAddToLiquidity = 9 * 10**12 * 10**9;
363
364   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
365   event SwapAndLiquifyEnabledUpdated(bool enabled);
366
```

# SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED
LINE 544

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
543    return _amount.mul(__reflectFee).div(
544    10**2
545    );
546    }
547
548
```

# SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED
LINE 550

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
549    return _amount.mul(__liquidityFee).div(
550    10**2
551    );
552    }
553
554
```

# SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED
LINE 557

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
556    return _amount.mul(operations_fee).div(
557    10**2
558    );
559    }
560
561
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED
LINE 597

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
596    require(amount > 0, "Transfer amount must be greater than zero");
597    if ((contract_deployed + 30 days) < block.timestamp) {
598    __liquidityFee = 0;
599    __prev_liquidityFee = 0;
600    }
601
```

# SWC-103 | A FLOATING PRAGMA IS SET.

LINE 6

## low SEVERITY

The current pragma Solidity directive is ""^0.6.12"". 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

- YetiCoin.sol

## Locations

```
5    // SPDX-License-Identifier: Unlicensed
6    pragma solidity ^0.6.12;
7
8    /**
9    * @dev Wrappers over Solidity's arithmetic operations with added overflow
10
```

# SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 358

## 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

- YetiCoin.sol

## Locations

```
357
358    bool inSwapAndLiquify;
359    bool public swapAndLiquifyEnabled = true;
360
361    uint256 public _maxWalletHolding = 25 * 10**12 * 10**9;
362
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS
LINE 497

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
496   uint256[4] memory tValues = _getTValuesArray(tAmount);
497   uint256[3] memory rValues = _getRValuesArray(tAmount, tValues[1], tValues[2],
tValues[3]);
498   return (rValues[0], rValues[1], rValues[2], tValues[0], tValues[1], tValues[2],
tValues[3]);
499   }
500
501
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS
LINE 497

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
496   uint256[4] memory tValues = _getTValuesArray(tAmount);
497   uint256[3] memory rValues = _getRValuesArray(tAmount, tValues[1], tValues[2],
tValues[3]);
498   return (rValues[0], rValues[1], rValues[2], tValues[0], tValues[1], tValues[2],
tValues[3]);
499   }
500
501
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS
LINE 497

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
496   uint256[4] memory tValues = _getTValuesArray(tAmount);
497   uint256[3] memory rValues = _getRValuesArray(tAmount, tValues[1], tValues[2],
tValues[3]);
498   return (rValues[0], rValues[1], rValues[2], tValues[0], tValues[1], tValues[2],
tValues[3]);
499   }
500
501
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS
LINE 498

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
497    uint256[3] memory rValues = _getRValuesArray(tAmount, tValues[1], tValues[2],
tValues[3]);
498    return (rValues[0], rValues[1], rValues[2], tValues[0], tValues[1], tValues[2],
tValues[3]);
499    }
500
501    function _getTValuesArray(uint256 tAmount) private view returns (uint256[4] memory
val) {
502
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS
LINE 498

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

## Source File
- YetiCoin.sol

## Locations

```
497   uint256[3] memory rValues = _getRValuesArray(tAmount, tValues[1], tValues[2],
tValues[3]);
498   return (rValues[0], rValues[1], rValues[2], tValues[0], tValues[1], tValues[2],
tValues[3]);
499   }
500
501   function _getTValuesArray(uint256 tAmount) private view returns (uint256[4] memory
val) {
502
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS
LINE 498

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
497   uint256[3] memory rValues = _getRValuesArray(tAmount, tValues[1], tValues[2],
tValues[3]);
498   return (rValues[0], rValues[1], rValues[2], tValues[0], tValues[1], tValues[2],
tValues[3]);
499   }
500
501   function _getTValuesArray(uint256 tAmount) private view returns (uint256[4] memory
val) {
502
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS
LINE 498

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
497   uint256[3] memory rValues = _getRValuesArray(tAmount, tValues[1], tValues[2],
tValues[3]);
498   return (rValues[0], rValues[1], rValues[2], tValues[0], tValues[1], tValues[2],
tValues[3]);
499   }
500
501   function _getTValuesArray(uint256 tAmount) private view returns (uint256[4] memory
val) {
502
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS
LINE 498

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
497   uint256[3] memory rValues = _getRValuesArray(tAmount, tValues[1], tValues[2],
tValues[3]);
498   return (rValues[0], rValues[1], rValues[2], tValues[0], tValues[1], tValues[2],
tValues[3]);
499   }
500
501   function _getTValuesArray(uint256 tAmount) private view returns (uint256[4] memory
val) {
502
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS
LINE 498

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
497   uint256[3] memory rValues = _getRValuesArray(tAmount, tValues[1], tValues[2],
tValues[3]);
498   return (rValues[0], rValues[1], rValues[2], tValues[0], tValues[1], tValues[2],
tValues[3]);
499   }
500
501   function _getTValuesArray(uint256 tAmount) private view returns (uint256[4] memory
val) {
502
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS
LINE 498

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
497   uint256[3] memory rValues = _getRValuesArray(tAmount, tValues[1], tValues[2],
tValues[3]);
498   return (rValues[0], rValues[1], rValues[2], tValues[0], tValues[1], tValues[2],
tValues[3]);
499   }
500
501   function _getTValuesArray(uint256 tAmount) private view returns (uint256[4] memory
val) {
502
```

SYSFIXED

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 661

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
660   address[] memory path = new address[](2);
661   path[0] = address(this);
662   path[1] = uniswapV2Router.WETH();
663
664   _approve(address(this), address(uniswapV2Router), tokenAmount);
665
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS
LINE 662

## low SEVERITY

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

## Source File

- YetiCoin.sol

## Locations

```
661   path[0] = address(this);
662   path[1] = uniswapV2Router.WETH();
663
664   _approve(address(this), address(uniswapV2Router), tokenAmount);
665
666
```

# 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.