



Bitindi Chain  
Smart Contract  
Audit Report

# TABLE OF CONTENTS

## Audited Details

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

## Summary

- Contract Summary
- Audit Findings Summary
- Vulnerabilities Summary

## Conclusion

## Audit Results

## Smart Contract Analysis

- Detected Vulnerabilities

## Disclaimer

## About Us

# AUDITED DETAILS

## Audited Project

Project name	Token ticker	Blockchain
Bitindi Chain	BNI	Binance Smart Chain

## Addresses

Contract address	0x77fc65deda64f0cca9e3aea7b9d8521f4151882e
Contract deployer address	0xFB1E1d8b25Ab32F3353C23f8420B68A0376d5083

## Project Website

<https://bitindi.com/>

## Codebase

<https://bscscan.com/address/0x77fc65deda64f0cca9e3aea7b9d8521f4151882e#code>

# SUMMARY

Bitindi Chain (Bitindi) is a decentralized, high-efficiency, and energy-saving layer-1 public chain. It is compatible with smart contracts and supports high-performance transactions. The endogenous token of Bitindi is \$BNI and it adopts the PoS consensus mechanism. Bitindi will continue to onboard billions of users with ultra-fast transactions, tiny fees, easy-to-use apps, and environmentally friendliness.

## Contract Summary

### Documentation Quality

Bitindi Chain 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 Bitindi Chain 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 118, 119, 121, 161, 162, 165 and 176.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 134, 134, 158, 158, 159, 159, 163, 163, 325, 353, 424, 431, 431, 431, 431, 432, 432, 436, 436, 436, 437, 437, 441, 441, 445, 445, 449, 449, 453, 453, 454, 454, 456, 456, 457, 458, 474, 474, 479, 479, 479, 479, 480, 480, 525, 526, 556, 570, 570, 630, 630, 631, 631, 646, 651, 704, 704, 706, 710, 715, 716, 716, 717 and 717.
- 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 586, 587, 716, 717 and 717.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 514.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 625.

# CONCLUSION

We have audited the Bitindi Chain project released on October 2022 to discover issues and identify potential security vulnerabilities in Bitindi Chain 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 satisfactory results with low-risk issues.

The issues found in the Bitindi Chain 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. The current pragma Solidity directive is `">=0.6.00.9.0"`. 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. It is best practice to set the visibility of state variables explicitly. The default visibility for  `"_tOwned"` is internal. Other possible visibility settings are public and private. Use of `"tx.origin"` as a part of authorization control, 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.

# 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.	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
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	ISSUE FOUND
Write to Arbitrary Storage Location	SWC-124	The contract is responsible for ensuring that only authorized user or contract accounts may write to sensitive storage locations.	PASS
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS
Insufficient Gas Griefing	SWC-126	Insufficient gas 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 <code>abi.encodePacked()</code> with multiple variable length arguments can, in certain situations, lead to a hash collision.	PASS
Hardcoded gas amount	SWC-134	The <code>transfer()</code> and <code>send()</code> functions forward a fixed amount of 2300 gas.	PASS
Unencrypted Private Data	SWC-136	It is a common misconception that private type variables cannot be read.	PASS



# SMART CONTRACT ANALYSIS

Started	Saturday Oct 08 2022 16:48:51 GMT+0000 (Coordinated Universal Time)
Finished	Sunday Oct 09 2022 12:28:49 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	BitindiChain.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-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-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-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 134

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
133  uint8 constant private _decimals = 18;
134  uint256 constant private _tTotal = startingSupply * (10 ** _decimals);
135
136  struct Fees {
137    uint16 buyFee;
138
```

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

LINE 134

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
133  uint8 constant private _decimals = 18;
134  uint256 constant private _tTotal = startingSupply * (10 ** _decimals);
135
136  struct Fees {
137    uint16 buyFee;
138
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 158

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
157
158  uint256 private _maxTxAmount = (_tTotal * 1) / 100;
159  uint256 private _maxWalletSize = (_tTotal * 1) / 100;
160
161  Cashier cashier;
162
```

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

LINE 158

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
157
158  uint256 private _maxTxAmount = (_tTotal * 1) / 100;
159  uint256 private _maxWalletSize = (_tTotal * 1) / 100;
160
161  Cashier cashier;
162
```



# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 159

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
158 uint256 private _maxTxAmount = (_tTotal * 1) / 100;
159 uint256 private _maxWalletSize = (_tTotal * 1) / 100;
160
161 Cashier cashier;
162 uint256 reflectorGas = 300000;
163
```

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

LINE 159

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
158 uint256 private _maxTxAmount = (_tTotal * 1) / 100;
159 uint256 private _maxWalletSize = (_tTotal * 1) / 100;
160
161 Cashier cashier;
162 uint256 reflectorGas = 300000;
163
```

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

LINE 163

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
162  uint256 reflectorGas = 300000;  
163  uint256 public minimumHoldForRewards = 10_000 * (10**_decimals);  
164  
165  bool inSwap;  
166  bool public contractSwapEnabled = false;  
167
```

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

LINE 163

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
162  uint256 reflectorGas = 300000;
163  uint256 public minimumHoldForRewards = 10_000 * (10**_decimals);
164
165  bool inSwap;
166  bool public contractSwapEnabled = false;
167
```

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

LINE 325

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
324   if (_allowances[sender][msg.sender] != type(uint256).max) {  
325     _allowances[sender][msg.sender] -= amount;  
326   }  
327  
328   return _transfer(sender, recipient, amount);  
329
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 353

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
352  if (timeSinceLastPair != 0) {
353  require(block.timestamp - timeSinceLastPair > 3 days, "3 Day cooldown.");
354  }
355  lpPairs[pair] = true;
356  timeSinceLastPair = block.timestamp;
357
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 424

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
423     require(transferFee <= maxTransferTaxes, "Cannot exceed maximums.");
424     require(buyFee + sellFee <= maxRoundtripTax, "Cannot exceed roundtrip maximum.");
425     _taxRates.buyFee = buyFee;
426     _taxRates.sellFee = sellFee;
427     _taxRates.transferFee = transferFee;
428
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 431

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
430     function setMaxTxPercent(uint256 percent, uint256 divisor) external onlyOwner {
431         require((_tTotal * percent) / divisor >= (_tTotal * 5 / 1000), "Max Transaction amt
must be above 0.5% of total supply.");
432         _maxTxAmount = (_tTotal * percent) / divisor;
433     }
434
435
```



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

LINE 431

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
430     function setMaxTxPercent(uint256 percent, uint256 divisor) external onlyOwner {
431         require((_tTotal * percent) / divisor >= (_tTotal * 5 / 1000), "Max Transaction amt
must be above 0.5% of total supply.");
432         _maxTxAmount = (_tTotal * percent) / divisor;
433     }
434
435
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 431

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
430     function setMaxTxPercent(uint256 percent, uint256 divisor) external onlyOwner {
431         require((_tTotal * percent) / divisor >= (_tTotal * 5 / 1000), "Max Transaction amt
must be above 0.5% of total supply.");
432         _maxTxAmount = (_tTotal * percent) / divisor;
433     }
434
435
```

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

LINE 431

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
430     function setMaxTxPercent(uint256 percent, uint256 divisor) external onlyOwner {
431         require((_tTotal * percent) / divisor >= (_tTotal * 5 / 1000), "Max Transaction amt
must be above 0.5% of total supply.");
432         _maxTxAmount = (_tTotal * percent) / divisor;
433     }
434
435
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 432

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
431   require((_tTotal * percent) / divisor >= (_tTotal * 5 / 1000), "Max Transaction amt
must be above 0.5% of total supply.");
432   _maxTxAmount = (_tTotal * percent) / divisor;
433   }
434
435   function setMaxWalletSize(uint256 percent, uint256 divisor) external onlyOwner {
436
```

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

LINE 432

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
431   require((_tTotal * percent) / divisor >= (_tTotal * 5 / 1000), "Max Transaction amt
must be above 0.5% of total supply.");
432   _maxTxAmount = (_tTotal * percent) / divisor;
433   }
434
435   function setMaxWalletSize(uint256 percent, uint256 divisor) external onlyOwner {
436
```

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

LINE 436

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
435     function setMaxWalletSize(uint256 percent, uint256 divisor) external onlyOwner {
436         require((_tTotal * percent) / divisor >= (_tTotal / 100), "Max Wallet amt must be
above 1% of total supply.");
437         _maxWalletSize = (_tTotal * percent) / divisor;
438     }
439
440
```

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

LINE 436

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
435     function setMaxWalletSize(uint256 percent, uint256 divisor) external onlyOwner {
436         require((_tTotal * percent) / divisor >= (_tTotal / 100), "Max Wallet amt must be
above 1% of total supply.");
437         _maxWalletSize = (_tTotal * percent) / divisor;
438     }
439
440
```

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

LINE 436

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
435     function setMaxWalletSize(uint256 percent, uint256 divisor) external onlyOwner {
436         require((_tTotal * percent) / divisor >= (_tTotal / 100), "Max Wallet amt must be
above 1% of total supply.");
437         _maxWalletSize = (_tTotal * percent) / divisor;
438     }
439
440
```



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

LINE 437

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
436   require((_tTotal * percent) / divisor >= (_tTotal / 100), "Max Wallet amt must be
above 1% of total supply.");
437   _maxWalletSize = (_tTotal * percent) / divisor;
438   }
439
440   function getMaxTX() public view returns (uint256) {
441
```

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

LINE 437

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
436   require((_tTotal * percent) / divisor >= (_tTotal / 100), "Max Wallet amt must be
above 1% of total supply.");
437   _maxWalletSize = (_tTotal * percent) / divisor;
438   }
439
440   function getMaxTX() public view returns (uint256) {
441
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 441

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
440     function getMaxTX() public view returns (uint256) {
441         return _maxTxAmount / (10**_decimals);
442     }
443
444     function getMaxWallet() public view returns (uint256) {
445
```

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

LINE 441

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
440 function getMaxTX() public view returns (uint256) {  
441     return _maxTxAmount / (10**_decimals);  
442 }  
443  
444 function getMaxWallet() public view returns (uint256) {  
445
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 445

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
444     function getMaxWallet() public view returns (uint256) {
445         return _maxWalletSize / (10**_decimals);
446     }
447
448     function getTokenAmountAtPriceImpact(uint256 priceImpactInHundreds) external view
returns (uint256) {
449
```

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

LINE 445

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
444     function getMaxWallet() public view returns (uint256) {
445         return _maxWalletSize / (10**_decimals);
446     }
447
448     function getTokenAmountAtPriceImpact(uint256 priceImpactInHundreds) external view
returns (uint256) {
449
```

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

LINE 449

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
448     function getTokenAmountAtPriceImpact(uint256 priceImpactInHundreds) external view
returns (uint256) {
449     return((balanceOf(lpPair) * priceImpactInHundreds) / masterTaxDivisor);
450 }
451
452     function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
453
```

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

LINE 449

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
448     function getTokenAmountAtPriceImpact(uint256 priceImpactInHundreds) external view
returns (uint256) {
449     return((balanceOf(lpPair) * priceImpactInHundreds) / masterTaxDivisor);
450 }
451
452     function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
453
```



# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 453

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
452 function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
453     swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
454     swapAmount = (_tTotal * amountPercent) / amountDivisor;
455     require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
456     require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
457
```

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

LINE 453

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
452 function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
453     swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
454     swapAmount = (_tTotal * amountPercent) / amountDivisor;
455     require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
456     require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
457
```

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

LINE 454

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
453 swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
454 swapAmount = (_tTotal * amountPercent) / amountDivisor;
455 require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
456 require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
457 require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
458
```

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

LINE 454

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
453 swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
454 swapAmount = (_tTotal * amountPercent) / amountDivisor;
455 require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
456 require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
457 require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
458
```

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

LINE 456

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
455     require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
456     require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
457     require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
458     require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
459     }
460
```

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

LINE 456

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
455     require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
456     require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
457     require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
458     require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
459     }
460
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 457

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
456   require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
457   require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
458   require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
459   }
460
461
```

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

LINE 458

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
457     require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
458     require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
459     }
460
461     function setPriceImpactSwapAmount(uint256 priceImpactSwapPercent) external
onlyOwner {
462
```



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

LINE 474

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
473  function setRewardsProperties(uint256 _minPeriod, uint256 _minReflection, uint256
minReflectionMultiplier) external onlyOwner {
474  _minReflection = _minReflection * 10**minReflectionMultiplier;
475  cashier.setRewardsProperties(_minPeriod, _minReflection);
476  }
477
478
```

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

LINE 474

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
473     function setRewardsProperties(uint256 _minPeriod, uint256 _minReflection, uint256
minReflectionMultiplier) external onlyOwner {
474         _minReflection = _minReflection * 10**minReflectionMultiplier;
475         cashier.setRewardsProperties(_minPeriod, _minReflection);
476     }
477
478
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 479

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
478     function setMinimumHoldForRewards(uint256 percent, uint256 divisor) external
onlyOwner {
479     require((_tTotal * percent) / divisor < (_tTotal * 2) / 100, "Cannot exceed maximum
amount for this value.");
480     minimumHoldForRewards = (_tTotal * percent) / divisor;
481 }
482
483
```

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

LINE 479

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
478     function setMinimumHoldForRewards(uint256 percent, uint256 divisor) external
onlyOwner {
479     require((_tTotal * percent) / divisor < (_tTotal * 2) / 100, "Cannot exceed maximum
amount for this value.");
480     minimumHoldForRewards = (_tTotal * percent) / divisor;
481 }
482
483
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 479

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
478     function setMinimumHoldForRewards(uint256 percent, uint256 divisor) external
onlyOwner {
479     require((_tTotal * percent) / divisor < (_tTotal * 2) / 100, "Cannot exceed maximum
amount for this value.");
480     minimumHoldForRewards = (_tTotal * percent) / divisor;
481 }
482
483
```

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

LINE 479

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
478     function setMinimumHoldForRewards(uint256 percent, uint256 divisor) external
onlyOwner {
479     require((_tTotal * percent) / divisor < (_tTotal * 2) / 100, "Cannot exceed maximum
amount for this value.");
480     minimumHoldForRewards = (_tTotal * percent) / divisor;
481 }
482
483
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 480

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
479     require((_tTotal * percent) / divisor < (_tTotal * 2) / 100, "Cannot exceed maximum
amount for this value.");
480     minimumHoldForRewards = (_tTotal * percent) / divisor;
481 }
482
483 function setReflectorSettings(uint256 gas) external onlyOwner {
484
```

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

LINE 480

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
479     require((_tTotal * percent) / divisor < (_tTotal * 2) / 100, "Cannot exceed maximum
amount for this value.");
480     minimumHoldForRewards = (_tTotal * percent) / divisor;
481 }
482
483 function setReflectorSettings(uint256 gas) external onlyOwner {
484
```



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

LINE 525

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
524 function _basicTransfer(address from, address to, uint256 amount) internal returns
    (bool) {
525     _tOwned[from] -= amount;
526     _tOwned[to] += amount;
527     emit Transfer(from, to, amount);
528     return true;
529
```

# SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 526

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
525  _tOwned[from] -= amount;  
526  _tOwned[to] += amount;  
527  emit Transfer(from, to, amount);  
528  return true;  
529  }  
530
```

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

LINE 556

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
555     if (!_isExcludedFromLimits[to]) {
556         require(balanceOf(to) + amount <= _maxWalletSize, "Transfer amount exceeds the
maxWalletSize.");
557     }
558 }
559 }
560
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 570

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
569  uint256 swapAmt = swapAmount;
570  if (piContractSwapsEnabled) { swapAmt = (balanceOf(lpPair) * piSwapPercent) /
masterTaxDivisor; }
571  if (contractTokenBalance >= swapAmt) { contractTokenBalance = swapAmt; }
572  contractSwap(contractTokenBalance);
573  }
574
```

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

LINE 570

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
569     uint256 swapAmt = swapAmount;
570     if (piContractSwapsEnabled) { swapAmt = (balanceOf(lpPair) * piSwapPercent) /
masterTaxDivisor; }
571     if (contractTokenBalance >= swapAmt) { contractTokenBalance = swapAmt; }
572     contractSwap(contractTokenBalance);
573 }
574
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 630

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
629     allowedPresaleExclusion = false;
630     swapThreshold = (balanceOf(lpPair) * 10) / 10000;
631     swapAmount = (balanceOf(lpPair) * 30) / 10000;
632     }
633
634
```

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

LINE 630

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
629     allowedPresaleExclusion = false;
630     swapThreshold = (balanceOf(lpPair) * 10) / 10000;
631     swapAmount = (balanceOf(lpPair) * 30) / 10000;
632   }
633
634
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 631

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
630     swapThreshold = (balanceOf(lpPair) * 10) / 10000;
631     swapAmount = (balanceOf(lpPair) * 30) / 10000;
632 }
633
634     function finalizeTransfer(address from, address to, uint256 amount, bool buy, bool
sell, bool other) internal returns (bool) {
635
```



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

LINE 631

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
630     swapThreshold = (balanceOf(lpPair) * 10) / 10000;
631     swapAmount = (balanceOf(lpPair) * 30) / 10000;
632   }
633
634   function finalizeTransfer(address from, address to, uint256 amount, bool buy, bool
sell, bool other) internal returns (bool) {
635
```

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

LINE 646

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
645
646  _tOwned[from] -= amount;
647  uint256 amountReceived = amount;
648  if (takeFee) {
649    amountReceived = takeTaxes(from, amount, buy, sell, other);
650
```

# SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 651

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
650     }
651     _tOwned[to] += amountReceived;
652     emit Transfer(from, to, amountReceived);
653     if (!_hasLiqBeenAdded) {
654         _checkLiquidityAdd(from, to);
655     }
```

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

LINE 704

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
703     || block.chainid == 56)) { currentFee = 4500; }
704     uint256 feeAmount = amount * currentFee / masterTaxDivisor;
705     if (feeAmount > 0) {
706         _tOwned[address(this)] += feeAmount;
707         emit Transfer(from, address(this), feeAmount);
708     }
```

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

LINE 704

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
703     || block.chainid == 56)) { currentFee = 4500; }
704     uint256 feeAmount = amount * currentFee / masterTaxDivisor;
705     if (feeAmount > 0) {
706         _tOwned[address(this)] += feeAmount;
707         emit Transfer(from, address(this), feeAmount);
708     }
```

## SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 706

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
705     if (feeAmount > 0) {  
706         _tOwned[address(this)] += feeAmount;  
707         emit Transfer(from, address(this), feeAmount);  
708     }  
709  
710
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 710

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
709
710     return amount - feeAmount;
711     }
712
713     function multiSendTokens(address[] memory accounts, uint256[] memory amounts)
external onlyOwner {
714
```

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

LINE 715

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
714   require(accounts.length == amounts.length, "Lengths do not match.");
715   for (uint16 i = 0; i < accounts.length; i++) {
716     require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
717     finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
718   }
719
```



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

LINE 716

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
715   for (uint16 i = 0; i < accounts.length; i++) {
716     require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
717     finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
718   }
719 }
720
```

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

LINE 716

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
715     for (uint16 i = 0; i < accounts.length; i++) {
716         require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
717         finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
718     }
719 }
720
```

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

LINE 717

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
716     require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
717     finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
718   }
719 }
720
721
```

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

LINE 717

## low SEVERITY

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

## Source File

- BitindiChain.sol

## Locations

```
716     require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
717     finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
718   }
719 }
720
721
```

## 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

- BitindiChain.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 118

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
117 contract BitindiChain is IERC20 {
118     mapping (address => uint256) _tOwned;
119     mapping (address => bool) lpPairs;
120     uint256 private timeSinceLastPair = 0;
121     mapping (address => mapping (address => uint256)) _allowances;
122 }
```

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

LINE 119

### 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

- BitindiChain.sol

### Locations

```
118 mapping (address => uint256) _tOwned;
119 mapping (address => bool) lpPairs;
120 uint256 private timeSinceLastPair = 0;
121 mapping (address => mapping (address => uint256)) _allowances;
122 mapping (address => bool) private _isExcludedFromProtection;
123
```

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

LINE 121

### 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

- BitindiChain.sol

### Locations

```
120 uint256 private timeSinceLastPair = 0;
121 mapping (address => mapping (address => uint256)) _allowances;
122 mapping (address => bool) private _isExcludedFromProtection;
123 mapping (address => bool) private _isExcludedFromFees;
124 mapping (address => bool) private _isExcludedFromLimits;
125
```



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

LINE 161

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
160
161  Cashier cashier;
162  uint256 reflectorGas = 300000;
163  uint256 public minimumHoldForRewards = 10_000 * (10**_decimals);
164
165
```

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

LINE 162

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
161  Cashier cashier;  
162  uint256 reflectorGas = 300000;  
163  uint256 public minimumHoldForRewards = 10_000 * (10**_decimals);  
164  
165  bool inSwap;  
166
```

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

LINE 165

### 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

- BitindiChain.sol

### Locations

```
164
165  bool inSwap;
166  bool public contractSwapEnabled = false;
167  uint256 public swapThreshold;
168  uint256 public swapAmount;
169
```

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

LINE 176

### 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

- BitindiChain.sol

### Locations

```
175  bool public _hasLiqBeenAdded = false;
176  Protections protections;
177
178  modifier inSwapFlag() {
179    inSwap = true;
180  }
```

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

LINE 514

## 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

- BitindiChain.sol

## Locations

```
513    && to != _owner
514    && tx.origin != _owner
515    && !_liquidityHolders[to]
516    && !_liquidityHolders[from]
517    && to != DEAD
518
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 586

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
585     address[] memory path = new address[](2);
586     path[0] = address(this);
587     path[1] = dexRouter.WETH();
588
589     try dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(
590
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 587

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
586 path[0] = address(this);
587 path[1] = dexRouter.WETH();
588
589 try dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(
590 contractTokenBalance,
591
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 716

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
715   for (uint16 i = 0; i < accounts.length; i++) {  
716     require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");  
717     finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,  
true);  
718   }  
719 }  
720
```



## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 717

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
716     require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
717     finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
718   }
719 }
720
721
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 717

### low SEVERITY

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

### Source File

- BitindiChain.sol

### Locations

```
716     require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
717     finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
718   }
719 }
720
721
```

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

LINE 625

### 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

- BitindiChain.sol

### Locations

```
624  }
625  try protections.setLaunch(lpPair, uint32(block.number), uint64(block.timestamp),
_decimals) {} catch {}
626  try cashier.initialize() {} catch {}
627  tradingEnabled = true;
628  processReflect = true;
629
```

# 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.