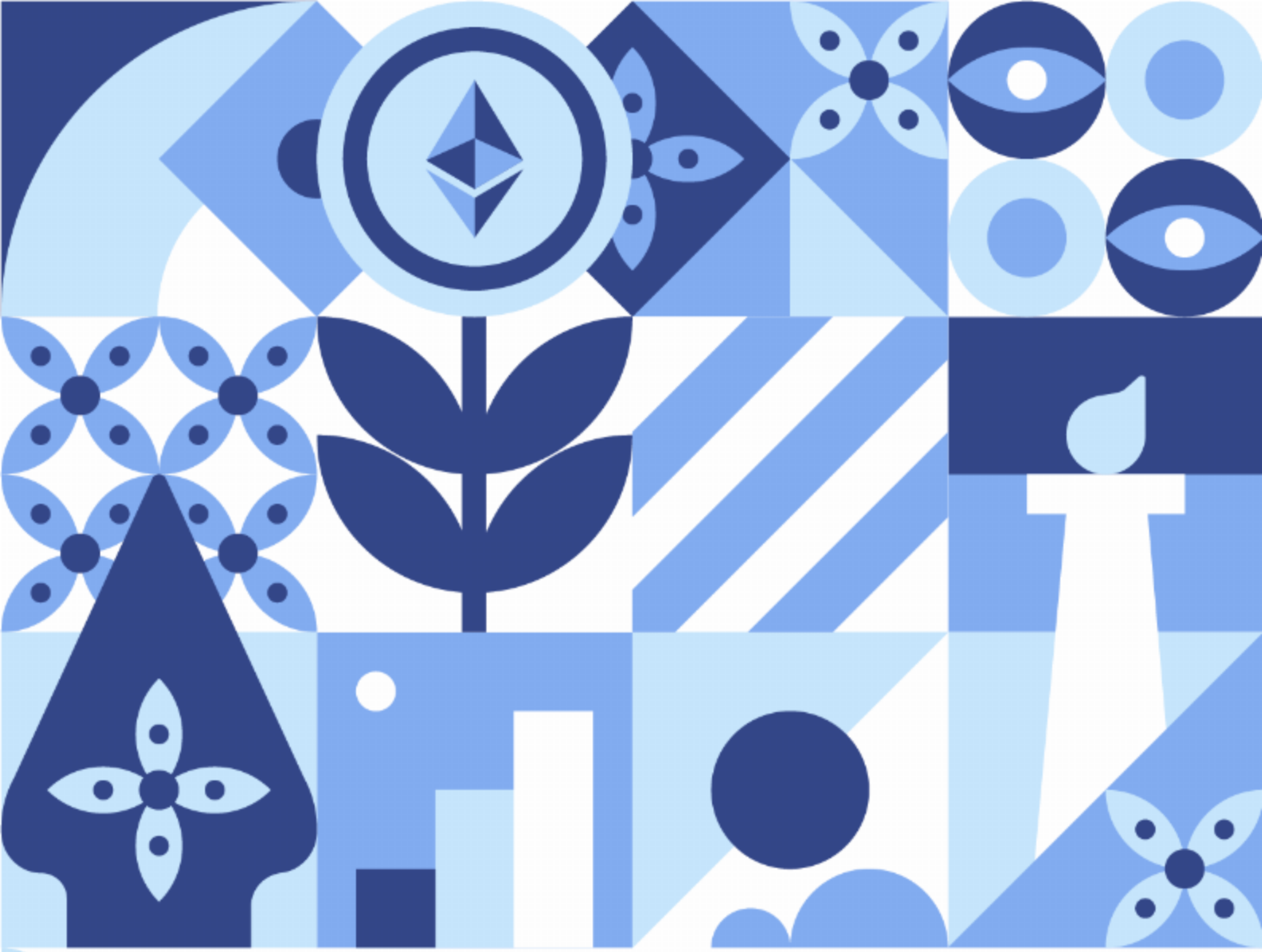

BARFIGHT

Smart Contract Audit Report

SYSFIXED

21 Sep 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
BARFIGHT	BFIGHT	Ethereum

Addresses

Contract address	0x6b91b72931993449fecC9d590D0d786a41588b1E
Contract deployer address	0xBE5f757b4c1dd913e9cD3a0CaD0F68C3DE62b6dD

Project Website

https://barfight.io/

Codebase

https://etherscan.io/address/0x6b91b72931993449fecC9d590D0d786a41588b1E#code

SUMMARY

$BFIGHT token is set to take the metaverse market by storm. This is virtual barfights! Design your patron's attire whether it be a wife beater singlet or suit, select your weapon of choice from bottles to tasers and get swinging – winner takes all!

Contract Summary

Documentation Quality

BARFIGHT 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 BARFIGHT 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 1093 and 1101.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 274, 286, 303, 304, 320, 334, 348, 362, 375, 384, 400, 421, 445, 476, 861, 877, 902, 935, 936, 957, 958, 981, 983, 1093, 1093, 1093, 1093, 1095, 1095, 1095, 1097, 1099, 1099, 1099, 1099, 1100, 1100, 1100, 1100, 1180, 1180, 1180, 1181, 1247, 1290, 1303, 1328, 1328, 1332, 1332, 1333, 1334, 1337, 1339, 1364, 1365, 1370, 1407 and 1421.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 25, 80, 106, 202, 254, 477, 508, 592, 677, 703 and 1070.
- 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 1248, 1473 and 1474.

CONCLUSION

We have audited the BARFIGHT project released on September 2022 to discover issues and identify potential security vulnerabilities in BARFIGHT 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 BARFIGHT 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 Sep 20 2022 16:23:39 GMT+0000 (Coordinated Universal Time)
Finished	Wednesday Sep 21 2022 01:39:20 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	BFIGHT.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-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	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-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 274

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
273   if (b > a) return (false, 0);
274   return (true, a - b);
275   }
276   }
277
278
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 286

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
285   // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
286   // benefit is lost if 'b' is also tested.
287   // See: https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
288   if (a == 0) return (true, 0);
289   uint256 c = a * b;
290
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 303

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
302   if (b == 0) return (false, 0);
303   return (true, a / b);
304   }
305   }
306
307
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 304

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
303   return (true, a / b);
304   }
305   }
306
307   /**
308
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 320

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
319   /**
320   * @dev Returns the addition of two unsigned integers, reverting on
321   * overflow.
322   *
323   * Counterpart to Solidity's `+` operator.
324
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 334

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
333   /**
334   * @dev Returns the subtraction of two unsigned integers, reverting on
335   * overflow (when the result is negative).
336   *
337   * Counterpart to Solidity's `-` operator.
338
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 348

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
347   /**
348   * @dev Returns the multiplication of two unsigned integers, reverting on
349   * overflow.
350   *
351   * Counterpart to Solidity's `*` operator.
352
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 362

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
361   /**
362   * @dev Returns the integer division of two unsigned integers, reverting on
363   * division by zero. The result is rounded towards zero.
364   *
365   * Counterpart to Solidity's `/` operator.
366
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 375

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
374
375   /**
376   * @dev Returns the remainder of dividing two unsigned integers. (unsigned integer
modulo),
377   * reverting when dividing by zero.
378   *
379
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 384

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
383   * Requirements:
384   *
385   * - The divisor cannot be zero.
386   */
387   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
388
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 400

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
399   *
400   * Requirements:
401   *
402   * - Subtraction cannot overflow.
403   */
404
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 421

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
420   * `revert` opcode (which leaves remaining gas untouched) while Solidity
421   * uses an invalid opcode to revert (consuming all remaining gas).
422   *
423   * Requirements:
424   *
425
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 445

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
444   *
445   * Counterpart to Solidity's `%` operator. This function uses a `revert`
446   * opcode (which leaves remaining gas untouched) while Solidity uses an
447   * invalid opcode to revert (consuming all remaining gas).
448   *
449
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 476

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
475   * via msg.sender and msg.data, they should not be accessed in such a direct
476   * manner, since when dealing with meta-transactions the account sending and
477   * paying for execution may not be the actual sender (as far as an application
478   * is concerned).
479   *
480
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 861

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
860   *
861   * - `spender` cannot be the zero address.
862   */
863   function increaseAllowance(address spender, uint256 addedValue) public virtual
returns (bool) {
864   _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
865
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 877

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
876   * Requirements:
877   *
878   * - `spender` cannot be the zero address.
879   * - `spender` must have allowance for the caller of at least
880   * `subtractedValue`.
881
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 902

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
901   *
902   * - `sender` cannot be the zero address.
903   * - `recipient` cannot be the zero address.
904   * - `sender` must have a balance of at least `amount`.
905   */
906
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 935

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
934   *
935   * - `account` cannot be the zero address.
936   */
937   function _mint(address account, uint256 amount) internal virtual {
938   require(account != address(0), "ERC20: mint to the zero address");
939
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 936

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
935   * - `account` cannot be the zero address.
936   */
937   function _mint(address account, uint256 amount) internal virtual {
938   require(account != address(0), "ERC20: mint to the zero address");
939
940
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 957

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
956   *
957   * - `account` cannot be the zero address.
958   * - `account` must have at least `amount` tokens.
959   */
960   function _burn(address account, uint256 amount) internal virtual {
961
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 958

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
957   * - `account` cannot be the zero address.
958   * - `account` must have at least `amount` tokens.
959   */
960   function _burn(address account, uint256 amount) internal virtual {
961   require(account != address(0), "ERC20: burn from the zero address");
962
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 981

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
980   * This internal function is equivalent to `approve`, and can be used to
981   * e.g. set automatic allowances for certain subsystems, etc.
982   *
983   * Emits an {Approval} event.
984   *
985
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 983

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
982   *
983   * Emits an {Approval} event.
984   *
985   * Requirements:
986   *
987
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1093

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1092
1093   address payable marketingWallet =
payable(address(0x393d8d86E8b5aB2D653A3A94364FC40f9DB97a65)); //  MARKETING WALLET
1094
1095
1096
1097
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1093

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1092
1093   address payable marketingWallet =
payable(address(0x393d8d86E8b5aB2D653A3A94364FC40f9DB97a65)); //  MARKETING WALLET
1094
1095
1096
1097
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1093

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1092
1093   address payable marketingWallet =
payable(address(0x393d8d86E8b5aB2D653A3A94364FC40f9DB97a65)); //  MARKETING WALLET
1094
1095
1096
1097
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1093

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1092
1093   address payable marketingWallet =
payable(address(0x393d8d86E8b5aB2D653A3A94364FC40f9DB97a65)); //  MARKETING WALLET
1094
1095
1096
1097
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1095

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1094
1095
1096
1097
1098   mapping(address => bool) private _isExcludedFromFees;
1099
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1095

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1094
1095
1096
1097
1098   mapping(address => bool) private _isExcludedFromFees;
1099
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1095

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1094
1095
1096
1097
1098   mapping(address => bool) private _isExcludedFromFees;
1099
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1097

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1096
1097
1098   mapping(address => bool) private _isExcludedFromFees;
1099   mapping(address => bool) private _isExcludedFromLimit;
1100   mapping(address => bool) public _isBlackListed;
1101
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1099

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1098   mapping(address => bool) private _isExcludedFromFees;
1099   mapping(address => bool) private _isExcludedFromLimit;
1100   mapping(address => bool) public _isBlackListed;
1101   bool isTradingEnabled;
1102
1103
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1099

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1098   mapping(address => bool) private _isExcludedFromFees;
1099   mapping(address => bool) private _isExcludedFromLimit;
1100   mapping(address => bool) public _isBlackListed;
1101   bool isTradingEnabled;
1102
1103
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1099

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1098   mapping(address => bool) private _isExcludedFromFees;
1099   mapping(address => bool) private _isExcludedFromLimit;
1100   mapping(address => bool) public _isBlackListed;
1101   bool isTradingEnabled;
1102
1103
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1099

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1098   mapping(address => bool) private _isExcludedFromFees;
1099   mapping(address => bool) private _isExcludedFromLimit;
1100   mapping(address => bool) public _isBlackListed;
1101   bool isTradingEnabled;
1102
1103
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1100

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1099   mapping(address => bool) private _isExcludedFromLimit;
1100   mapping(address => bool) public _isBlackListed;
1101   bool isTradingEnabled;
1102
1103   // store addresses that a automatic market maker pairs. Any transfer *to* these
addresses
1104
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1100

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1099   mapping(address => bool) private _isExcludedFromLimit;
1100   mapping(address => bool) public _isBlackListed;
1101   bool isTradingEnabled;
1102
1103   // store addresses that a automatic market maker pairs. Any transfer *to* these
addresses
1104
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1100

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1099   mapping(address => bool) private _isExcludedFromLimit;
1100   mapping(address => bool) public _isBlackListed;
1101   bool isTradingEnabled;
1102
1103   // store addresses that a automatic market maker pairs. Any transfer *to* these
addresses
1104
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1100

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1099   mapping(address => bool) private _isExcludedFromLimit;
1100   mapping(address => bool) public _isBlackListed;
1101   bool isTradingEnabled;
1102
1103   // store addresses that a automatic market maker pairs. Any transfer *to* these
addresses
1104
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1180

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1179   require(newAddress != address(uniswapV2Router), "TOKEN: The router already has
that address");
1180   uniswapV2Router = IUniswapV2Router02(newAddress);
1181   address get_pair =
1182   IUniswapV2Factory(uniswapV2Router.factory()).getPair(address(this),
1183   uniswapV2Router.WETH());
1184
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1180

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1179   require(newAddress != address(uniswapV2Router), "TOKEN: The router already has
that address");
1180   uniswapV2Router = IUniswapV2Router02(newAddress);
1181   address get_pair =
1182   IUniswapV2Factory(uniswapV2Router.factory()).getPair(address(this),
1183   uniswapV2Router.WETH());
1184
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1180

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1179   require(newAddress != address(uniswapV2Router), "TOKEN: The router already has
that address");
1180   uniswapV2Router = IUniswapV2Router02(newAddress);
1181   address get_pair =
1182   IUniswapV2Factory(uniswapV2Router.factory()).getPair(address(this),
1183   uniswapV2Router.WETH());
1184
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1181

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1180   uniswapV2Router = IUniswapV2Router02(newAddress);
1181   address get_pair =
1182   IUniswapV2Factory(uniswapV2Router.factory()).getPair(address(this),
1183   uniswapV2Router.WETH());
1184   if (get_pair == address(0)) {
1185
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1247

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1246   );
1247
1248   _setAutomatedMarketMakerPair(pair, value);
1249   }
1250
1251
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1290

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1289   totalSellFee = marketing + liquidity;
1290   require (totalSellFee <= 5, "max fees should be less than equal to 5%");
1291   }
1292
1293   function enableTrading () external onlyOwner {
1294
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1303

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1302   function setMarketingWallet(address newWallet) external onlyOwner {
1303   require (newWallet != (address(0)), "marketing wallet can't be a zero address");
1304   marketingWallet = payable(newWallet);
1305   }
1306
1307
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1328

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1327   function _transfer(
1328   address from,
1329   address to,
1330   uint256 amount
1331   ) internal override {
1332
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1328

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1327   function _transfer(
1328   address from,
1329   address to,
1330   uint256 amount
1331   ) internal override {
1332
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1332

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1331   ) internal override {
1332   require(from != address(0), "Token: transfer from the zero address");
1333   require(to != address(0), "Token: transfer to the zero address");
1334   require(
1335   !_isBlackListed[from] && !_isBlackListed[to],
1336
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1332

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1331   ) internal override {
1332   require(from != address(0), "Token: transfer from the zero address");
1333   require(to != address(0), "Token: transfer to the zero address");
1334   require(
1335   !_isBlackListed[from] && !_isBlackListed[to],
1336
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1333

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1332   require(from != address(0), "Token: transfer from the zero address");
1333   require(to != address(0), "Token: transfer to the zero address");
1334   require(
1335   !_isBlackListed[from] && !_isBlackListed[to],
1336   "Account is blacklisted"
1337
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1334

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1333   require(to != address(0), "Token: transfer to the zero address");
1334   require(
1335   !_isBlackListed[from] && !_isBlackListed[to],
1336   "Account is blacklisted"
1337   );
1338
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1337

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1336   "Account is blacklisted"
1337   );
1338
1339   if (amount == 0) {
1340   super._transfer(from, to, 0);
1341
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1339

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1338
1339   if (amount == 0) {
1340   super._transfer(from, to, 0);
1341   return;
1342   }
1343
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1364

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1363
1364   // if any account belongs to _isExcludedFromFee account then remove the fee
1365   if (_isExcludedFromFees[from] || _isExcludedFromFees[to]) {
1366   takeFee = false;
1367   }
1368
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1365

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1364   // if any account belongs to _isExcludedFromFee account then remove the fee
1365   if (_isExcludedFromFees[from] || _isExcludedFromFees[to]) {
1366   takeFee = false;
1367   }
1368
1369
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1370

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1369   if (takeFee) {
1370   require (isTradingEnabled, "Trading is not enabled yet");
1371   uint256 fees;
1372
1373
1374
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1407

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1406
1407   function swapAndSendToMarketing(uint256 tokens) private lockTheSwap {
1408   uint256 oldbalance = address(this).balance;
1409   swapTokensForEth(tokens);
1410   uint256 newBalance = address(this).balance - oldbalance;
1411
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1421

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1420
1421   // capture the contract's current ETH balance.
1422   // this is so that we can capture exactly the amount of ETH that the
1423   // swap creates, and not make the liquidity event include any ETH that
1424   // has been manually sent to the contract
1425
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 25

low SEVERITY

The current pragma Solidity directive is ""^0.5.0"". 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

- BFIGHT.sol

Locations

```
24   function decimals() external pure returns (uint8);
25   function totalSupply() external view returns (uint);
26   function balanceOf(address owner) external view returns (uint);
27   function allowance(address owner, address spender) external view returns (uint);
28
29
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 80

low SEVERITY

The current pragma Solidity directive is ""^0.5.0"". 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

- BFIGHT.sol

Locations

```
79   function getPair(address tokenA, address tokenB) external view returns (address
pair);
80   function allPairs(uint) external view returns (address pair);
81   function allPairsLength() external view returns (uint);
82
83   function createPair(address tokenA, address tokenB) external returns (address pair);
84
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 106

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- BFIGHT.sol

Locations

```
105   uint deadline
106   ) external returns (uint amountA, uint amountB, uint liquidity);
107   function addLiquidityETH(
108   address token,
109   uint amountTokenDesired,
110
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 202

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- BFIGHT.sol

Locations

```
201   function removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(
202   address token,
203   uint liquidity,
204   uint amountTokenMin,
205   uint amountETHMin,
206
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 254

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". 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

- BFIGHT.sol

Locations

```
253   /**
254   * @dev Returns the addition of two unsigned integers, with an overflow flag.
255   *
256   * _Available since v3.4._
257   */
258
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 477

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". 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

- BFIGHT.sol

Locations

```
476   * manner, since when dealing with meta-transactions the account sending and
477   * paying for execution may not be the actual sender (as far as an application
478   * is concerned).
479   *
480   * This contract is only required for intermediate, library-like contracts.
481
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 508

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". 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

- BFIGHT.sol

Locations

```
507   *
508   * This module is used through inheritance. It will make available the modifier
509   * `onlyOwner`, which can be applied to your functions to restrict their use to
510   * the owner.
511   */
512
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 592

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". 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

- BFIGHT.sol

Locations

```
591   /**
592   * @dev Moves `amount` tokens from the caller's account to `recipient`.
593   *
594   * Returns a boolean value indicating whether the operation succeeded.
595   *
596
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 677

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". 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

- BFIGHT.sol

Locations

```
676   */
677   function symbol() external view returns (string memory);
678
679   /**
680   * @dev Returns the decimals places of the token.
681
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 703

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". 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

- BFIGHT.sol

Locations

```
702   * TIP: For a detailed writeup see our guide
703   * https://forum.zeppelin.solutions/t/how-to-implement-erc20-supply-
mechanisms/226[How
704   * to implement supply mechanisms].
705   *
706   * We have followed general OpenZeppelin Contracts guidelines: functions revert
707
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1070

low SEVERITY

The current pragma Solidity directive is ""^0.8.10"". 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

- BFIGHT.sol

Locations

```
1069   uint16 marketingFee;
1070   uint16 liquidityFee;
1071
1072   }
1073
1074
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1093

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1092
1093   address payable marketingWallet =
payable(address(0x393d8d86E8b5aB2D653A3A94364FC40f9DB97a65)); //  MARKETING WALLET
1094
1095
1096
1097
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1101

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1100   mapping(address => bool) public _isBlackListed;
1101   bool isTradingEnabled;
1102
1103   // store addresses that a automatic market maker pairs. Any transfer *to* these
addresses
1104   // could be subject to a maximum transfer amount
1105
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1248

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1247
1248   _setAutomatedMarketMakerPair(pair, value);
1249   }
1250
1251   function _setAutomatedMarketMakerPair(address pair, bool value) private {
1252
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1473

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1472
1473
1474    }
1475
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1474

low SEVERITY

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

Source File

- BFIGHT.sol

Locations

```
1473
1474    }
1475
```

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.