FIT Token

Smart Contract Audit Report

SYSFIXED

03 May 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
FIT Token	FIT	Binance Smart Chain

Addresses

Contract address	0x77922a521182a719a48ba650ac2a040269888888
Contract deployer address	0xad80314c566Be4Bacbb3992F844faE914D9Fd31d

Project Website

https://calo.run/

Codebase

https://bscscan.com/address/0x77922a521182a719a48ba650ac2a040269888888#contracts

SUMMARY

Calo Metaverse blockchain system provides you chances to work out on a daily basis either in single or world challenge mode. We also record your training results and convert your moving movement into valuable rewards. You can either hold the Tokens and NFTs earned to use in-app or cash out for profit. The more you practice, the more rewards you get; that is the motivation that pushes us forward to move our bodies so as to gain both beneficial health and passive income

Contract Summary

Documentation Quality

FIT Token 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 FIT Token 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 1531, 1532, 1533, 1536 and 1537.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 35, 53, 72, 73, 90, 106, 121, 135, 149, 163, 179, 202, 225, 251, 281, 282, 286, 287, 287, 288, 303, 317, 317, 320, 320, 320, 1264, 1294, 1330, 1332, 1353, 1354, 1379, 1381, 1433, 1542, 1549, 1622 and 1631.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 11, 260, 333, 438, 465, 500, 661, 687, 944, 1035, 1063, 1483 and 1519.
- 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 287, 318, 319, 321, 321, 1623 and 1632.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 1559, 1560 and 1561.

CONCLUSION

We have audited the FIT Token project released on May 2022 to discover issues and identify potential security vulnerabilities in the FIT Token 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 FIT Token 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, use of "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 an invalid array index value. The current pragma Solidity directive is ""^0.8.0"". Specifying a fixed compiler version is recommended 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 "buyFeeRate" is internal. Other possible visibility settings are public and private. Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you know what you are doing.

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.	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	Monday May 02 2022 01:39:20 GMT+0000 (Coordinated Universal Time)
Finished	Tuesday May 03 2022 20:04:13 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	FITToken.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-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-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-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-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	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-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 35

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
34   unchecked {
35   uint256 c = a + b;
36   if (c < a) return (false, 0);
37   return (true, c);
38   }
39
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 53

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
52   if (b > a) return (false, 0);
53   return (true, a - b);
54   }
55   }
56
57
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 72

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
71   if (a == 0) return (true, 0);
72   uint256 c = a * b;
73   if (c / a != b) return (false, 0);
74   return (true, c);
75   }
76
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 73

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
72   uint256 c = a * b;
73   if (c / a != b) return (false, 0);
74   return (true, c);
75   }
76   }
77
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 90

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
89   if (b == 0) return (false, 0);
90   return (true, a / b);
91   }
92   }
93
94
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 106

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
105   if (b == 0) return (false, 0);
106   return (true, a % b);
107   }
108   }
109
110
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 121

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
120   function add(uint256 a, uint256 b) internal pure returns (uint256) {
121   return a + b;
122   }
123
124   /**
125
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 135

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
134   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
135   return a - b;
136   }
137
138   /**
139
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 149

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
148   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
149   return a * b;
150   }
151
152   /**
153
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 163

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
162   function div(uint256 a, uint256 b) internal pure returns (uint256) {
163   return a / b;
164   }
165
166   /**
167
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 179

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
178   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
179   return a % b;
180   }
181
182   /**
183
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 202

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
201   require(b <= a, errorMessage);
202   return a - b;
203   }
204   }
205
206
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 225

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
224   require(b > 0, errorMessage);
225   return a / b;
226   }
227   }
228
229
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 251

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
250   require(b > 0, errorMessage);
251   return a % b;
252   }
253   }
254   }
255
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 281

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
280   while (temp != 0) {
281   digits++;
282   temp /= 10;
283   }
284   bytes memory buffer = new bytes(digits);
285
```

SWC-101 | ARITHMETIC OPERATION "/=" DISCOVERED

LINE 282

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
281   digits++;
282   temp /= 10;
283   }
284   bytes memory buffer = new bytes(digits);
285   while (value != 0) {
286
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 286

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
285   while (value != 0) {
286   digits -= 1;
287   buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
288   value /= 10;
289   }
290
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 287

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
286   digits -= 1;
287   buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
288   value /= 10;
289   }
290   return string(buffer);
291
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 287

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
286   digits -= 1;
287   buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
288   value /= 10;
289   }
290   return string(buffer);
291
```

SWC-101 | ARITHMETIC OPERATION "/=" DISCOVERED

LINE 288

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
287   buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
288   value /= 10;
289   }
290   return string(buffer);
291   }
292
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 303

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
302   while (temp != 0) {
303   length++;
304   temp >>= 8;
305   }
306   return toHexString(value, length);
307
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 317

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
316   {
317   bytes memory buffer = new bytes(2 * length + 2);
318   buffer[0] = "0";
319   buffer[1] = "x";
320   for (uint256 i = 2 * length + 1; i > 1; --i) {
321
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 317

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
316   {
317   bytes memory buffer = new bytes(2 * length + 2);
318   buffer[0] = "0";
319   buffer[1] = "x";
320   for (uint256 i = 2 * length + 1; i > 1; --i) {
321
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 320

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
319   buffer[1] = "x";
320   for (uint256 i = 2 * length + 1; i > 1; --i) {
321   buffer[i] = _HEX_SYMBOLS[value & 0xf];
322   value >>= 4;
323   }
324
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 320

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
319   buffer[1] = "x";
320   for (uint256 i = 2 * length + 1; i > 1; --i) {
321   buffer[i] = _HEX_SYMBOLS[value & 0xf];
322   value >>= 4;
323   }
324
```

SWC-101 | ARITHMETIC OPERATION "--" DISCOVERED

LINE 320

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
319   buffer[1] = "x";
320   for (uint256 i = 2 * length + 1; i > 1; --i) {
321   buffer[i] = _HEX_SYMBOLS[value & 0xf];
322   value >>= 4;
323   }
324
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1264

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
1263   address owner = _msgSender();
1264   _approve(owner, spender, _allowances[owner][spender] + addedValue);
1265   return true;
1266   }
1267
1268
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1294

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
1293   unchecked {
1294   _approve(owner, spender, currentAllowance - subtractedValue);
1295   }
1296
1297   return true;
1298
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1330

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
1329   unchecked {
1330   _balances[from] = fromBalance - amount;
1331   }
1332   _balances[to] += amount;
1333
1334
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1332

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
1331   }
1332   _balances[to] += amount;
1333
1334   emit Transfer(from, to, amount);
1335
1336
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1353

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
1352
1353   _totalSupply += amount;
1354   _balances[account] += amount;
1355   emit Transfer(address(0), account, amount);
1356
1357
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1354

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
1353   _totalSupply += amount;
1354   _balances[account] += amount;
1355   emit Transfer(address(0), account, amount);
1356
1357   _afterTokenTransfer(address(0), account, amount);
1358
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1379

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
1378   unchecked {
1379   _balances[account] = accountBalance - amount;
1380   }
1381   _totalSupply -= amount;
1382
1383
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 1381

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
1380   }
1381   _totalSupply -= amount;
1382
1383   emit Transfer(account, address(0), amount);
1384
1385
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1433

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
1432   unchecked {
1433   _approve(owner, spender, currentAllowance - amount);
1434   }
1435   }
1436   }
1437
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1542

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
1541   require(
1542   lastSwapTimestamp[receipient] + antiBotTime < block.timestamp,
1543   "Anti front running bot"
1544   );
1545   lastSwapTimestamp[receipient] = block.timestamp;
1546
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1549

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
1548   require(
1549   lastSwapTimestamp[sender] + antiBotTime < block.timestamp,
1550   "Anti front running bot"
1551   );
1552   lastSwapTimestamp[sender] = block.timestamp;
1553
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1622

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
1621   {
1622   for (uint256 i = 0; i < addresses.length; i++) {
1623   lpAddresses[addresses[i]] = true;
1624   }
1625   }
1626
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1631

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
1630   {
1631   for (uint256 i = 0; i < addresses.length; i++) {
1632   blacklist[addresses[i]] = isBlacklist;
1633   }
1634   }
1635
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 11

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

- FITToken.sol

Locations

```
10
11   pragma solidity ^0.8.0;
12
13   // CAUTION
14   // This version of SafeMath should only be used with Solidity 0.8 or later,
15
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 260

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

- FITToken.sol

Locations

```
259
260   pragma solidity ^0.8.0;
261
262   /**
263   * @dev String operations.
264
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 333

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

- FITToken.sol

Locations

```
332
333   pragma solidity ^0.8.0;
334
335   /**
336   * @dev External interface of AccessControl declared to support ERC165 detection.
337
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 438

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

- FITToken.sol

Locations

```
437
438   pragma solidity ^0.8.0;
439
440   /**
441   * @dev Interface of the ERC165 standard, as defined in the
442
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 465

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

- FITToken.sol

Locations

```
464
465   pragma solidity ^0.8.0;
466
467   /**
468   * @dev Implementation of the {IERC165} interface.
469
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 500

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

- FITToken.sol

Locations

```
499
500   pragma solidity ^0.8.0;
501
502   /**
503   * @dev Required interface of an ERC721 compliant contract.
504
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 661

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

- FITToken.sol

Locations

```
660
661   pragma solidity ^0.8.0;
662
663   /**
664   * @dev Provides information about the current execution context, including the
665
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 687

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

- FITToken.sol

Locations

```
686
687   pragma solidity ^0.8.0;
688
689   /**
690   * @dev Contract module that allows children to implement role-based access
691
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 944

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

- FITToken.sol

Locations

```
943
944   pragma solidity ^0.8.0;
945
946   /**
947   * @dev Interface of the ERC20 standard as defined in the EIP.
948
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1035

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

- FITToken.sol

Locations

```
1034
1035   pragma solidity ^0.8.0;
1036
1037   /**
1038   * @dev Interface for the optional metadata functions from the ERC20 standard.
1039
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1063

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

- FITToken.sol

Locations

```
1062
1063   pragma solidity ^0.8.0;
1064
1065   /**
1066   * @dev Implementation of the {IERC20} interface.
1067
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1483

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

- FITToken.sol

Locations

```
1482
1483   pragma solidity ^0.8.0;
1484
1485   /**
1486   * @dev Extension of {ERC20} that allows token holders to destroy both their own
1487
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1519

low SEVERITY

The current pragma Solidity directive is ""^0.8.2"". 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

- FITToken.sol

Locations

```
1518
1519   pragma solidity ^0.8.2;
1520
1521   contract FITToken is ERC20Burnable, AccessControl {
1522   using SafeMath for uint256;
1523
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1531

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
1530
1531   uint256 buyFeeRate = 0;
1532   uint256 sellFeeRate = 0;
1533   address teamWallet;
1534   uint256 public antiBotTime = 30 seconds;
1535
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1532

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
1531   uint256 buyFeeRate = 0;
1532   uint256 sellFeeRate = 0;
1533   address teamWallet;
1534   uint256 public antiBotTime = 30 seconds;
1535   mapping(address => bool) public blacklist;
1536
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1533

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
1532   uint256 sellFeeRate = 0;
1533   address teamWallet;
1534   uint256 public antiBotTime = 30 seconds;
1535   mapping(address => bool) public blacklist;
1536   mapping(address => bool) lpAddresses;
1537
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1536

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
1535   mapping(address => bool) public blacklist;
1536   mapping(address => bool) lpAddresses;
1537   mapping(address => uint256) lastSwapTimestamp;
1538
1539   modifier antiFrontRunning(address sender, address receipient) {
1540
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1537

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
1536   mapping(address => bool) lpAddresses;
1537   mapping(address => uint256) lastSwapTimestamp;
1538
1539   modifier antiFrontRunning(address sender, address receipient) {
1540   if (lpAddresses[sender] == true) {
1541
```

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

LINE 1559

low SEVERITY

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source File

- FITToken.sol

Locations

```
1558   constructor() ERC20("FIT Token", "FIT") {
1559   _setupRole(DEFAULT_ADMIN_ROLE, tx.origin);
1560   _setupRole(MINTER_ROLE, tx.origin);
1561   teamWallet = tx.origin;
1562   //create FIT/WBNB pair in Pancake swap
1563
```

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

LINE 1560

low SEVERITY

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source File

- FITToken.sol

Locations

```
1559   _setupRole(DEFAULT_ADMIN_ROLE, tx.origin);
1560   _setupRole(MINTER_ROLE, tx.origin);
1561   teamWallet = tx.origin;
1562   //create FIT/WBNB pair in Pancake swap
1563   address pair = IPancakeFactory(PANCAKE_FACTORY_ADDRESS).createPair(
1564
```

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

LINE 1561

low SEVERITY

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source File

- FITToken.sol

Locations

```
1560   _setupRole(MINTER_ROLE, tx.origin);
1561   teamWallet = tx.origin;
1562   //create FIT/WBNB pair in Pancake swap
1563   address pair = IPancakeFactory(PANCAKE_FACTORY_ADDRESS).createPair(
1564   address(this),
1565
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 287

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
286   digits -= 1;
287   buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
288   value /= 10;
289   }
290   return string(buffer);
291
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 318

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
317   bytes memory buffer = new bytes(2 * length + 2);
318   buffer[0] = "0";
319   buffer[1] = "x";
320   for (uint256 i = 2 * length + 1; i > 1; --i) {
321   buffer[i] = _HEX_SYMBOLS[value & 0xf];
322
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 319

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
318   buffer[0] = "0";
319   buffer[1] = "x";
320   for (uint256 i = 2 * length + 1; i > 1; --i) {
321   buffer[i] = _HEX_SYMBOLS[value & 0xf];
322   value >>= 4;
323
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 321

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
320   for (uint256 i = 2 * length + 1; i > 1; --i) {
321   buffer[i] = _HEX_SYMBOLS[value & 0xf];
322   value >>= 4;
323   }
324   require(value == 0, "Strings: hex length insufficient");
325
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 321

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
320   for (uint256 i = 2 * length + 1; i > 1; --i) {
321   buffer[i] = _HEX_SYMBOLS[value & 0xf];
322   value >>= 4;
323   }
324   require(value == 0, "Strings: hex length insufficient");
325
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1623

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
1622   for (uint256 i = 0; i < addresses.length; i++) {
1623   lpAddresses[addresses[i]] = true;
1624   }
1625   }
1626
1627
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1632

low SEVERITY

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

Source File

- FITToken.sol

Locations

```
1631   for (uint256 i = 0; i < addresses.length; i++) {
1632   blacklist[addresses[i]] = isBlacklist;
1633   }
1634   }
1635
1636
```

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.