ERC20

Smart Contract Audit Report

SYSFIXED

11 Mar 2021

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
ERC20	ERC20	Binance Smart Chain

Addresses

Contract address	0x58730ae0faa10d73b0cddb5e7b87c3594f7a20cb
Contract deployer address	0xE1fd5CDd0C0B2804B32ab94e03346c78826980AE

Project Website

https://erc20.tech/

Codebase

https://bscscan.com/address/0x58730ae0faa10d73b0cddb5e7b87c3594f7a20cb#code

SUMMARY

ERC20 Token — is a cryptocurrency, an open-source, public, blockchain-based coin with operating Ethereum Blockchain Network featuring smart contract functionality in the version of erc-20 standard protocol. Although the Ethereum Foundation and figureheads like Vitalik played a significant role in Ethereum's trajectory early on, the community is taking up the torch to decentralize processes. From these five years, Decision-making processes in Ethereum have matured significantly as the network and users have grown. The Ethereum Network and erc-20 standard became more popular each year through dissemination, use, and popularization. Having a bar is very important as it allows to coin to be compatible with every wallet and every exchange built to the same standards. The standards provide the functionality to transfer tokens, send and receive, and allow tokens to be approved to be another on-chain third party, etc., that can spend theming; Bitcoin could ship and store only by email, and on hard disks, there were no digital wallets. So we are sure it was one of the main problems in widely popularizing and distributing Bitcoin since it was not easy to store. Conclusion about erc standard: There is no way to confuse the erc-20 and ERC20 Token. It's an entirely different tool. But we are here to help users if they have any questions. ERC20 Token has registered in the Ethereum contract ABI language, and the contract source code of ERC20 and name are verified—many tokens on the blockchain use erc 20 standards. And there is plenty of different bars to choose from: ERC, ERC 20, ERC 137, ERC 681, IEEE, etc. And now, through the years, erc standard and ERC20 Token have been widely adopted (100.00 ERC20 Holders) because of the new users, popularization, and naming. Nowadays, some coins do not even have half of the advantages, integrations, and updates that the ERC20 Token now has.

Contract Summary

Documentation Quality

ERC20 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 ERC20 with the discovery of several low issues.

Test Coverage

Test coverage of the project is 100% (Through Codebase)

Audit Findings Summary

- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 130, 142, 155, 156, 167, 177, 191, 208, 223, 224, 242, 259, 277, 297, 317, 1288 and 1312.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 11, 38, 108, 325, 426, 736, 803, 809, 853, 880, 972, 1006, 1039, 1231, 1365, 1421, 1586, 1610 and 1631.
- 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 1289, 1289 and 1313.

CONCLUSION

We have audited the ERC20 project released on March 2021 to discover issues and identify potential security vulnerabilities in ERC20 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 ERC20 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, and out-of-bounds array access which the index access expression can cause an exception in case an invalid array index value is used. The current pragma Solidity directive is "">=0.6.00.8.0"". 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.

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.	PASS
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	Wednesday Mar 10 2021 20:14:42 GMT+0000 (Coordinated Universal Time)
Finished	Thursday Mar 11 2021 00:20:59 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	AmazingBEP20.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-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-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-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 130

low SEVERITY

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

Source File

- AmazingBEP20.sol

Locations

```
129   function tryAdd(uint256 a, uint256 b) internal pure returns (bool, uint256) {
130   uint256 c = a + b;
131   if (c < a) return (false, 0);
132   return (true, c);
133   }
134
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 142

low SEVERITY

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

Source File

- AmazingBEP20.sol

Locations

```
141   if (b > a) return (false, 0);
142   return (true, a - b);
143   }
144
145   /**
146
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 155

low SEVERITY

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

Source File

- AmazingBEP20.sol

Locations

```
154   if (a == 0) return (true, 0);
155   uint256 c = a * b;
156   if (c / a != b) return (false, 0);
157   return (true, c);
158   }
159
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 156

low SEVERITY

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

Source File

- AmazingBEP20.sol

Locations

```
155   uint256 c = a * b;
156   if (c / a != b) return (false, 0);
157   return (true, c);
158   }
159
160
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 167

low SEVERITY

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

Source File

- AmazingBEP20.sol

Locations

```
166   if (b == 0) return (false, 0);
167   return (true, a / b);
168   }
169
170   /**
171
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 177

low SEVERITY

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

Source File

- AmazingBEP20.sol

Locations

```
176   if (b == 0) return (false, 0);
177   return (true, a % b);
178   }
179
180   /**
181
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 191

low SEVERITY

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

Source File

- AmazingBEP20.sol

Locations

```
190   function add(uint256 a, uint256 b) internal pure returns (uint256) {
191   uint256 c = a + b;
192   require(c >= a, "SafeMath: addition overflow");
193   return c;
194   }
195
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 208

low SEVERITY

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

Source File

- AmazingBEP20.sol

Locations

```
207   require(b <= a, "SafeMath: subtraction overflow");
208   return a - b;
209   }
210
211   /**
212
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 223

low SEVERITY

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

Source File

- AmazingBEP20.sol

Locations

```
222   if (a == 0) return 0;
223   uint256 c = a * b;
224   require(c / a == b, "SafeMath: multiplication overflow");
225   return c;
226   }
227
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 224

low SEVERITY

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

Source File

- AmazingBEP20.sol

Locations

```
223   uint256 c = a * b;
224   require(c / a == b, "SafeMath: multiplication overflow");
225   return c;
226   }
227
228
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 242

low SEVERITY

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

Source File

- AmazingBEP20.sol

Locations

```
241   require(b > 0, "SafeMath: division by zero");
242   return a / b;
243   }
244
245   /**
246
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 259

low SEVERITY

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

Source File

- AmazingBEP20.sol

Locations

```
258   require(b > 0, "SafeMath: modulo by zero");
259   return a % b;
260   }
261
262   /**
263
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 277

low SEVERITY

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

Source File

- AmazingBEP20.sol

Locations

```
276   require(b <= a, errorMessage);
277   return a - b;
278   }
279
280   /**
281
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 297

low SEVERITY

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

Source File

- AmazingBEP20.sol

Locations

```
296   require(b > 0, errorMessage);
297   return a / b;
298   }
299
300   /**
301
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 317

low SEVERITY

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

Source File

- AmazingBEP20.sol

Locations

```
316   require(b > 0, errorMessage);
317   return a % b;
318   }
319   }
320
321
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1288

low SEVERITY

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

Source File

- AmazingBEP20.sol

Locations

```
1287   // query support of each interface in interfaceIds
1288   for (uint256 i = 0; i < interfaceIds.length; i++) {
1289   interfaceIdsSupported[i] = _supportsERC165Interface(account, interfaceIds[i]);
1290   }
1291   }
1292
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1312

low SEVERITY

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

Source File

- AmazingBEP20.sol

Locations

```
1311   // query support of each interface in _interfaceIds
1312   for (uint256 i = 0; i < interfaceIds.length; i++) {
1313   if (!_supportsERC165Interface(account, interfaceIds[i])) {
1314   return false;
1315   }
1316
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 11

low SEVERITY

The current pragma Solidity directive is ""=0.6.0<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

- AmazingBEP20.sol

Locations

```
10
11   pragma solidity >=0.6.0 <0.8.0;
12
13   /*
14   * @dev Provides information about the current execution context, including the
15
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 38

low SEVERITY

The current pragma Solidity directive is ""=0.6.0<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

- AmazingBEP20.sol

Locations

```
37
38   pragma solidity >=0.6.0 <0.8.0;
39
40   /**
41   * @dev Contract module which provides a basic access control mechanism, where
42
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 108

low SEVERITY

The current pragma Solidity directive is ""=0.6.0<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

- AmazingBEP20.sol

Locations

```
107
108   pragma solidity >=0.6.0 <0.8.0;
109
110   /**
111   * @dev Wrappers over Solidity's arithmetic operations with added overflow
112
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 325

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- AmazingBEP20.sol

Locations

```
324
325   pragma solidity ^0.7.0;
326
327   /**
328   * @dev Interface of the BEP standard.
329
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 426

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- AmazingBEP20.sol

Locations

```
425
426   pragma solidity ^0.7.0;
427
428
429
430
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 736

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- AmazingBEP20.sol

Locations

```
735
736   pragma solidity ^0.7.0;
737
738
739   /**
740
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 803

low SEVERITY

The current pragma Solidity directive is ""=0.6.0<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

- AmazingBEP20.sol

Locations

```
802
803   pragma solidity >=0.6.0 <0.8.0;
804
805   // File: contracts/token/BEP20/lib/BEP20Burnable.sol
806
807
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 809

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- AmazingBEP20.sol

Locations

```
808
809   pragma solidity ^0.7.0;
810
811
812
813
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 853

low SEVERITY

The current pragma Solidity directive is ""=0.6.0<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

- AmazingBEP20.sol

Locations

```
852
853   pragma solidity >=0.6.0 <0.8.0;
854
855   /**
856   * @dev Interface of the ERC165 standard, as defined in the
857
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 880

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- AmazingBEP20.sol

Locations

```
879
880   pragma solidity ^0.7.0;
881
882
883
884
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 972

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- AmazingBEP20.sol

Locations

```
971
972   pragma solidity ^0.7.0;
973
974   /**
975   * @title IBEP20OperableReceiver Interface
976
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1006

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- AmazingBEP20.sol

Locations

```
1005
1006   pragma solidity ^0.7.0;
1007
1008   /**
1009   * @title IBEP20OperableSpender Interface
1010
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1039

low SEVERITY

The current pragma Solidity directive is ""=0.6.2<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

- AmazingBEP20.sol

Locations

```
1038
1039   pragma solidity >=0.6.2 <0.8.0;
1040
1041   /**
1042   * @dev Collection of functions related to the address type
1043
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1231

low SEVERITY

The current pragma Solidity directive is ""=0.6.2<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

- AmazingBEP20.sol

Locations

```
1230
1231   pragma solidity >=0.6.2 <0.8.0;
1232
1233   /**
1234   * @dev Library used to query support of an interface declared via {IERC165}.
1235
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1365

low SEVERITY

The current pragma Solidity directive is ""=0.6.0<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

- AmazingBEP20.sol

Locations

```
1364
1365   pragma solidity >=0.6.0 <0.8.0;
1366
1367
1368   /**
1369
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1421

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- AmazingBEP20.sol

Locations

```
1420
1421   pragma solidity ^0.7.0;
1422
1423
1424
1425
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1586

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- AmazingBEP20.sol

Locations

```
1585
1586   pragma solidity ^0.7.0;
1587
1588
1589
1590
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1610

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- AmazingBEP20.sol

Locations

```
1609
1610   pragma solidity ^0.7.0;
1611
1612   interface IPayable {
1613   function pay(string memory serviceName) external payable;
1614
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1631

low SEVERITY

The current pragma Solidity directive is ""^0.7.0"". 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

- AmazingBEP20.sol

Locations

```
1630
1631   pragma solidity ^0.7.0;
1632
1633
1634
1635
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1289

low SEVERITY

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

Source File

- AmazingBEP20.sol

Locations

```
1288   for (uint256 i = 0; i < interfaceIds.length; i++) {
1289   interfaceIdsSupported[i] = _supportsERC165Interface(account, interfaceIds[i]);
1290   }
1291   }
1292
1293
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1289

low SEVERITY

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

Source File

- AmazingBEP20.sol

Locations

```
1288   for (uint256 i = 0; i < interfaceIds.length; i++) {
1289   interfaceIdsSupported[i] = _supportsERC165Interface(account, interfaceIds[i]);
1290   }
1291   }
1292
1293
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1313

low SEVERITY

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

Source File

- AmazingBEP20.sol

Locations

```
1312   for (uint256 i = 0; i < interfaceIds.length; i++) {
1313   if (!_supportsERC165Interface(account, interfaceIds[i])) {
1314   return false;
1315   }
1316   }
1317
```

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.