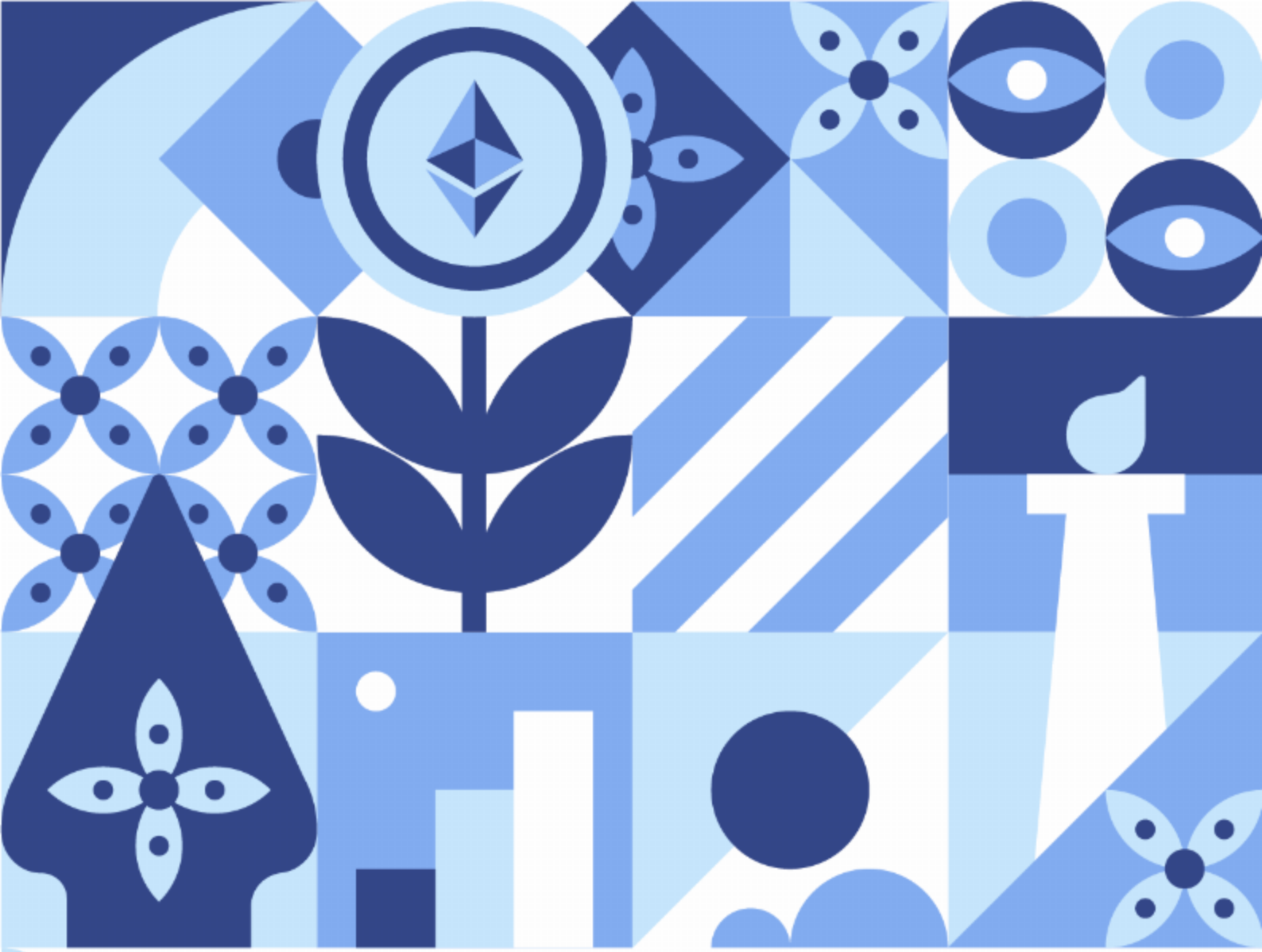

IMPT

Smart Contract Audit Report

SYSFIXED

04 Oct 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
IMPT	IMPT	Ethereum

Addresses

Contract address	0x04c17b9d3b29a78f7bd062a57cf44fc633e71f85
Contract deployer address	0xae500791254Bc813F336c3A1054e31ADe2b583F1

Project Website

https://www.impt.io/

Codebase

https://etherscan.io/address/0x04c17b9d3b29a78f7bd062a57cf44fc633e71f85#code

SUMMARY

Join an impactful carbon offset program by investing in the IMPT token.
Become a part of a large ecosystem that connects socially responsible brands with businesses and individuals who want to reduce their carbon footprint. Based on the blockchain, our platform empowers you to buy, sell, or retire carbon credits while avoiding double counting and fraud.

Contract Summary

Documentation Quality

IMPT 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 IMPT 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 95, 103, 139, 140, 144, 145, 145, 146, 161, 171, 171, 174, 174, 174, 299, 904, 927, 960, 962, 983, 984, 1009, 1011, 1060, 1381, 1382, 1384 and 103.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 71, 117, 195, 415, 521, 584, 669, 699, 726, 1111, 1208 and 1249.
- 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 145, 172, 173, 175, 175, 1383, 1384 and 1384.

CONCLUSION

We have audited the IMPT project released on October 2022 to discover issues and identify potential security vulnerabilities in IMPT 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 IMPT 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 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.	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	Monday Oct 03 2022 23:16:45 GMT+0000 (Coordinated Universal Time)
Finished	Tuesday Oct 04 2022 13:51:52 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	IMPT.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	COMPILER-REWRITABLE "<UINT> - 1" 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-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 95

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
94   unchecked {
95   counter._value += 1;
96   }
97   }
98
99
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 103

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
102   unchecked {
103   counter._value = value - 1;
104   }
105   }
106
107
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 139

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
138   while (temp != 0) {
139   digits++;
140   temp /= 10;
141   }
142   bytes memory buffer = new bytes(digits);
143
```

SWC-101 | ARITHMETIC OPERATION "/=" DISCOVERED

LINE 140

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
139   digits++;
140   temp /= 10;
141   }
142   bytes memory buffer = new bytes(digits);
143   while (value != 0) {
144
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 144

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
143   while (value != 0) {
144   digits -= 1;
145   buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
146   value /= 10;
147   }
148
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 145

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
144   digits -= 1;
145   buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
146   value /= 10;
147   }
148   return string(buffer);
149
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 145

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
144   digits -= 1;
145   buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
146   value /= 10;
147   }
148   return string(buffer);
149
```

SWC-101 | ARITHMETIC OPERATION "/=" DISCOVERED

LINE 146

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
145   buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
146   value /= 10;
147   }
148   return string(buffer);
149   }
150
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 161

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
160   while (temp != 0) {
161   length++;
162   temp >>= 8;
163   }
164   return toHexString(value, length);
165
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 171

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
170   function toHexString(uint256 value, uint256 length) internal pure returns (string
memory) {
171   bytes memory buffer = new bytes(2 * length + 2);
172   buffer[0] = "0";
173   buffer[1] = "x";
174   for (uint256 i = 2 * length + 1; i > 1; --i) {
175
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 171

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
170   function toHexString(uint256 value, uint256 length) internal pure returns (string
memory) {
171   bytes memory buffer = new bytes(2 * length + 2);
172   buffer[0] = "0";
173   buffer[1] = "x";
174   for (uint256 i = 2 * length + 1; i > 1; --i) {
175
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 174

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
173   buffer[1] = "x";
174   for (uint256 i = 2 * length + 1; i > 1; --i) {
175   buffer[i] = _HEX_SYMBOLS[value & 0xf];
176   value >>= 4;
177   }
178
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 174

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
173   buffer[1] = "x";
174   for (uint256 i = 2 * length + 1; i > 1; --i) {
175   buffer[i] = _HEX_SYMBOLS[value & 0xf];
176   value >>= 4;
177   }
178
```

SWC-101 | ARITHMETIC OPERATION "--" DISCOVERED

LINE 174

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
173   buffer[1] = "x";
174   for (uint256 i = 2 * length + 1; i > 1; --i) {
175   buffer[i] = _HEX_SYMBOLS[value & 0xf];
176   value >>= 4;
177   }
178
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 299

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
298   bytes32 s = vs &
bytes32(0x7fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff);
299   uint8 v = uint8((uint256(vs) >> 255) + 27);
300   return tryRecover(hash, v, r, s);
301   }
302
303
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 904

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
903   address owner = _msgSender();
904   _approve(owner, spender, allowance(owner, spender) + addedValue);
905   return true;
906   }
907
908
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 927

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
926   unchecked {
927   _approve(owner, spender, currentAllowance - subtractedValue);
928   }
929
930   return true;
931
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 960

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
959   unchecked {
960   _balances[from] = fromBalance - amount;
961   }
962   _balances[to] += amount;
963
964
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 962

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
961   }
962   _balances[to] += amount;
963
964   emit Transfer(from, to, amount);
965
966
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 983

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
982
983   _totalSupply += amount;
984   _balances[account] += amount;
985   emit Transfer(address(0), account, amount);
986
987
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 984

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
983   _totalSupply += amount;
984   _balances[account] += amount;
985   emit Transfer(address(0), account, amount);
986
987   _afterTokenTransfer(address(0), account, amount);
988
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1009

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
1008   unchecked {
1009   _balances[account] = accountBalance - amount;
1010   }
1011   _totalSupply -= amount;
1012
1013
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 1011

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
1010   }
1011   _totalSupply -= amount;
1012
1013   emit Transfer(account, address(0), amount);
1014
1015
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1060

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
1059   unchecked {
1060   _approve(owner, spender, currentAllowance - amount);
1061   }
1062   }
1063   }
1064
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1381

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
1380   require(recipients_.length <= 20, "Invalid recipients length");
1381   uint256 dec_ = 10**decimals();
1382   for (uint256 i = 0; i < recipients_.length; i++) {
1383   require(amounts_[i] > 0, "Amount is not positive");
1384   _mint(recipients_[i], amounts_[i] * dec_);
1385
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1382

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
1381   uint256 dec_ = 10**decimals();
1382   for (uint256 i = 0; i < recipients_.length; i++) {
1383   require(amounts_[i] > 0, "Amount is not positive");
1384   _mint(recipients_[i], amounts_[i] * dec_);
1385   }
1386
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1384

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
1383   require(amounts_[i] > 0, "Amount is not positive");
1384   _mint(recipients_[i], amounts_[i] * dec_);
1385   }
1386   }
1387
1388
```

SWC-101 | COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED

LINE 103

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
102   unchecked {
103   counter._value = value - 1;
104   }
105   }
106
107
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 71

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

- IMPT.sol

Locations

```
70
71   pragma solidity ^0.8.0;
72
73   /**
74   * @title Counters
75
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 117

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

- IMPT.sol

Locations

```
116
117   pragma solidity ^0.8.0;
118
119   /**
120   * @dev String operations.
121
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 195

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

- IMPT.sol

Locations

```
194
195   pragma solidity ^0.8.0;
196
197
198   /**
199
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 415

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

- IMPT.sol

Locations

```
414
415   pragma solidity ^0.8.0;
416
417
418   /**
419
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 521

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

- IMPT.sol

Locations

```
520
521   pragma solidity ^0.8.0;
522
523   /**
524   * @dev Interface of the ERC20 Permit extension allowing approvals to be made via
signatures, as defined in
525
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 584

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

- IMPT.sol

Locations

```
583
584   pragma solidity ^0.8.0;
585
586   /**
587   * @dev Interface of the ERC20 standard as defined in the EIP.
588
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 669

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

- IMPT.sol

Locations

```
668
669   pragma solidity ^0.8.0;
670
671
672   /**
673
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 699

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

- IMPT.sol

Locations

```
698
699   pragma solidity ^0.8.0;
700
701   /**
702   * @dev Provides information about the current execution context, including the
703
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 726

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

- IMPT.sol

Locations

```
725
726   pragma solidity ^0.8.0;
727
728
729
730
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1111

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

- IMPT.sol

Locations

```
1110
1111   pragma solidity ^0.8.0;
1112
1113
1114
1115
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1208

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

- IMPT.sol

Locations

```
1207
1208   pragma solidity ^0.8.0;
1209
1210
1211
1212
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1249

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

- IMPT.sol

Locations

```
1248
1249   pragma solidity ^0.8.0;
1250
1251
1252   /**
1253
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 145

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
144   digits -= 1;
145   buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
146   value /= 10;
147   }
148   return string(buffer);
149
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 172

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
171   bytes memory buffer = new bytes(2 * length + 2);
172   buffer[0] = "0";
173   buffer[1] = "x";
174   for (uint256 i = 2 * length + 1; i > 1; --i) {
175   buffer[i] = _HEX_SYMBOLS[value & 0xf];
176
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 173

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
172   buffer[0] = "0";
173   buffer[1] = "x";
174   for (uint256 i = 2 * length + 1; i > 1; --i) {
175   buffer[i] = _HEX_SYMBOLS[value & 0xf];
176   value >>= 4;
177
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 175

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
174   for (uint256 i = 2 * length + 1; i > 1; --i) {
175   buffer[i] = _HEX_SYMBOLS[value & 0xf];
176   value >>= 4;
177   }
178   require(value == 0, "Strings: hex length insufficient");
179
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 175

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
174   for (uint256 i = 2 * length + 1; i > 1; --i) {
175   buffer[i] = _HEX_SYMBOLS[value & 0xf];
176   value >>= 4;
177   }
178   require(value == 0, "Strings: hex length insufficient");
179
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1383

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
1382   for (uint256 i = 0; i < recipients_.length; i++) {
1383   require(amounts_[i] > 0, "Amount is not positive");
1384   _mint(recipients_[i], amounts_[i] * dec_);
1385   }
1386   }
1387
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1384

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
1383   require(amounts_[i] > 0, "Amount is not positive");
1384   _mint(recipients_[i], amounts_[i] * dec_);
1385   }
1386   }
1387
1388
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1384

low SEVERITY

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

Source File

- IMPT.sol

Locations

```
1383   require(amounts_[i] > 0, "Amount is not positive");
1384   _mint(recipients_[i], amounts_[i] * dec_);
1385   }
1386   }
1387
1388
```

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.