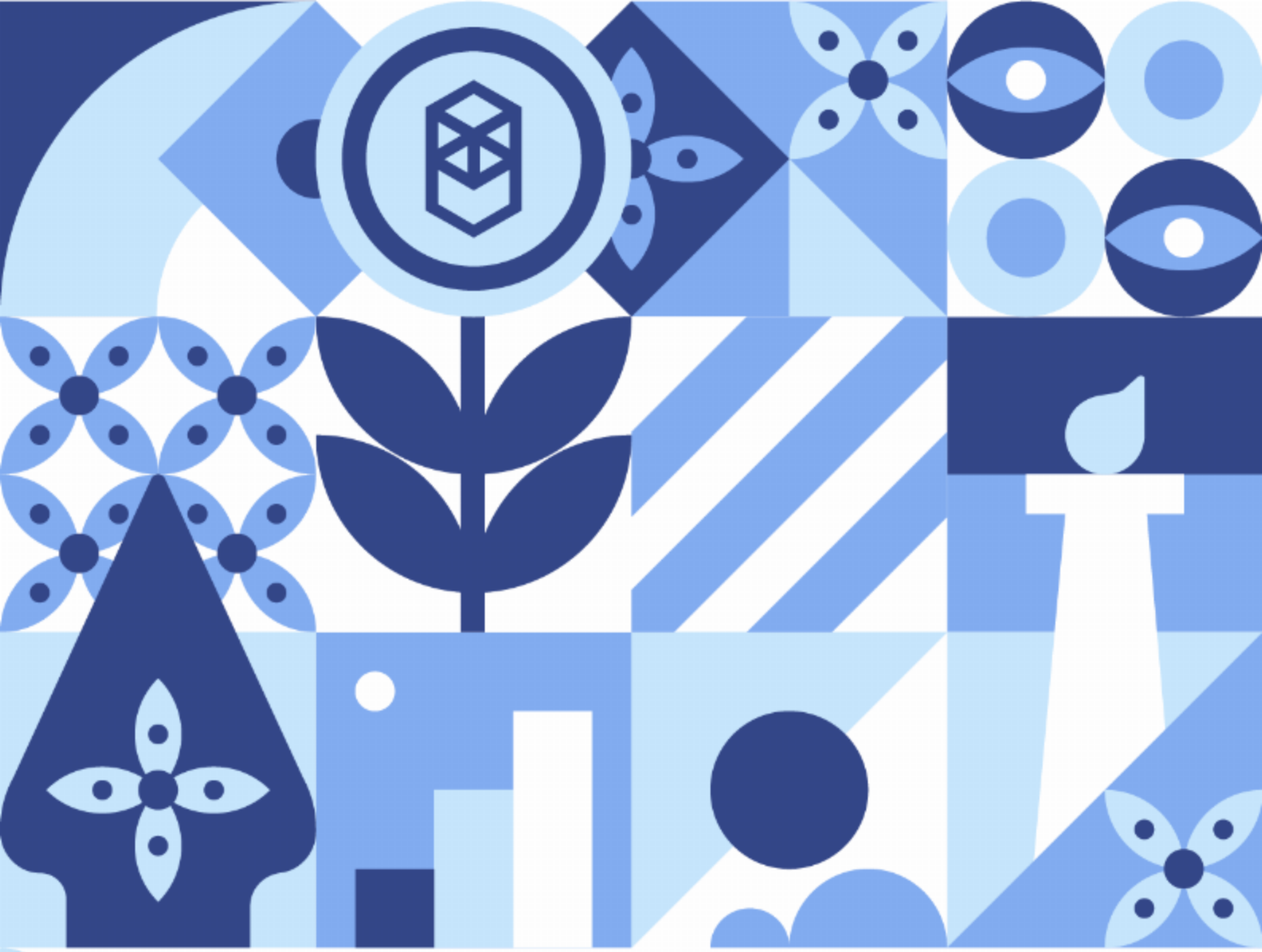

SCARAB

Smart Contract Audit Report

SYSFIXED

18 Dec 2021

TABLE OF CONTENTS

Audited Details

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

Summary

- Contract Summary
- Audit Findings Summary
- Vulnerabilities Summary

Conclusion

Audit Results

Smart Contract Analysis

- Detected Vulnerabilities

Disclaimer

About Us

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
SCARAB	SCARAB	Fantom

Addresses

Contract address	0x2e79205648b85485731cfe3025d66cf2d3b059c4
Contract deployer address	0x7abc0e9130c0B172f18Da62b02Eb65D136BD76E3

Project Website

https://twitter.com/Scarab_Finance

Codebase

https://ftmscan.com/address/0x2e79205648b85485731cfe3025d66cf2d3b059c4#code

SUMMARY

Scarab.Finance is a brand new Tomb finance fork on the Fantom Opera network.

Contract Summary

Documentation Quality

SCARAB 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 SCARAB 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 136, 148, 161, 162, 173, 183, 197, 214, 229, 230, 248, 265, 283, 303, 323, 710, 710, 710, 710, 710, 710, 710, 710, 746, 778, 801, 802, 837, 873, 1092, 1095, 1122 and 1092.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 5, 33, 114, 332, 640, 684, 882 and 889.
- 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 1041, 1042, 1092, 1095, 1097, 1104, 1123, 1124, 1125 and 1126.

CONCLUSION

We have audited the SCARAB project released on December 2021 to discover issues and identify potential security vulnerabilities in SCARAB 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 SCARAB 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, floating pragmas set on several lines, a public state variable with array type causing reachable exception by default 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	Friday Dec 17 2021 14:15:58 GMT+0000 (Coordinated Universal Time)
Finished	Saturday Dec 18 2021 06:52:49 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Scarab.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	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-110	PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.	low	acknowledged
SWC-110	PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.	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 136

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
135   function tryAdd(uint256 a, uint256 b) internal pure returns (bool, uint256) {
136   uint256 c = a + b;
137   if (c < a) return (false, 0);
138   return (true, c);
139   }
140
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 148

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
147   if (b > a) return (false, 0);
148   return (true, a - b);
149   }
150
151   /**
152
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 161

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
160   if (a == 0) return (true, 0);
161   uint256 c = a * b;
162   if (c / a != b) return (false, 0);
163   return (true, c);
164   }
165
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 162

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
161   uint256 c = a * b;
162   if (c / a != b) return (false, 0);
163   return (true, c);
164   }
165
166
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 173

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
172   if (b == 0) return (false, 0);
173   return (true, a / b);
174   }
175
176   /**
177
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 183

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
182   if (b == 0) return (false, 0);
183   return (true, a % b);
184   }
185
186   /**
187
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 197

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
196   function add(uint256 a, uint256 b) internal pure returns (uint256) {
197   uint256 c = a + b;
198   require(c >= a, "SafeMath: addition overflow");
199   return c;
200   }
201
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 214

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
213   require(b <= a, "SafeMath: subtraction overflow");
214   return a - b;
215   }
216
217   /**
218
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 229

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
228   if (a == 0) return 0;
229   uint256 c = a * b;
230   require(c / a == b, "SafeMath: multiplication overflow");
231   return c;
232   }
233
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 230

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
229   uint256 c = a * b;
230   require(c / a == b, "SafeMath: multiplication overflow");
231   return c;
232   }
233
234
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 248

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
247   require(b > 0, "SafeMath: division by zero");
248   return a / b;
249   }
250
251   /**
252
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 265

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
264   require(b > 0, "SafeMath: modulo by zero");
265   return a % b;
266   }
267
268   /**
269
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 283

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
282   require(b <= a, errorMessage);
283   return a - b;
284   }
285
286   /**
287
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 303

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
302   require(b > 0, errorMessage);
303   return a / b;
304   }
305
306   /**
307
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 323

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
322   require(b > 0, errorMessage);
323   return a % b;
324   }
325   }
326
327
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 710

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
709   // (a + b) / 2 can overflow, so we distribute
710   return (a / 2) + (b / 2) + ((a % 2 + b % 2) / 2);
711   }
712   }
713
714
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 710

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
709   // (a + b) / 2 can overflow, so we distribute
710   return (a / 2) + (b / 2) + ((a % 2 + b % 2) / 2);
711   }
712   }
713
714
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 710

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
709   // (a + b) / 2 can overflow, so we distribute
710   return (a / 2) + (b / 2) + ((a % 2 + b % 2) / 2);
711   }
712   }
713
714
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 710

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
709   // (a + b) / 2 can overflow, so we distribute
710   return (a / 2) + (b / 2) + ((a % 2 + b % 2) / 2);
711   }
712   }
713
714
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 710

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
709   // (a + b) / 2 can overflow, so we distribute
710   return (a / 2) + (b / 2) + ((a % 2 + b % 2) / 2);
711   }
712   }
713
714
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 710

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
709   // (a + b) / 2 can overflow, so we distribute
710   return (a / 2) + (b / 2) + ((a % 2 + b % 2) / 2);
711   }
712   }
713
714
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 710

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
709   // (a + b) / 2 can overflow, so we distribute
710   return (a / 2) + (b / 2) + ((a % 2 + b % 2) / 2);
711   }
712   }
713
714
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 710

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
709   // (a + b) / 2 can overflow, so we distribute
710   return (a / 2) + (b / 2) + ((a % 2 + b % 2) / 2);
711   }
712   }
713
714
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 746

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
745   function add(uint8 a, uint8 b) internal pure returns (uint8) {
746   uint8 c = a + b;
747   require(c >= a, "SafeMath: addition overflow");
748
749   return c;
750
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 778

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
777   require(b <= a, errorMessage);
778   uint8 c = a - b;
779
780   return c;
781   }
782
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 801

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
800
801   uint8 c = a * b;
802   require(c / a == b, "SafeMath: multiplication overflow");
803
804   return c;
805
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 802

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
801   uint8 c = a * b;
802   require(c / a == b, "SafeMath: multiplication overflow");
803
804   return c;
805   }
806
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 837

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
836   require(b > 0, errorMessage);
837   uint8 c = a / b;
838   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
839
840   return c;
841
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 873

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
872   require(b != 0, errorMessage);
873   return a % b;
874   }
875   }
876
877
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1092

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
1091   if (_index > 0) {
1092   require(_value > taxTiersTwaps[_index - 1]);
1093   }
1094   if (_index < getTaxTiersTwapsCount().sub(1)) {
1095   require(_value < taxTiersTwaps[_index + 1]);
1096
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1095

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
1094   if (_index < getTaxTiersTwapsCount().sub(1)) {
1095   require(_value < taxTiersTwaps[_index + 1]);
1096   }
1097   taxTiersTwaps[_index] = _value;
1098   return true;
1099
```

SWC-101 | ARITHMETIC OPERATION "--" DISCOVERED

LINE 1122

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
1121   if (autoCalculateTax) {
1122   for (uint8 tierId = uint8(getTaxTiersTwapsCount()).sub(1); tierId >= 0; --tierId)
{
1123   if (_tombPrice >= taxTiersTwaps[tierId]) {
1124   require(taxTiersRates[tierId] < 10000, "tax equal or bigger to 100%");
1125   taxRate = taxTiersRates[tierId];
1126
```

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

LINE 1092

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
1091   if (_index > 0) {
1092   require(_value > taxTiersTwaps[_index - 1]);
1093   }
1094   if (_index < getTaxTiersTwapsCount().sub(1)) {
1095   require(_value < taxTiersTwaps[_index + 1]);
1096
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 5

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

- Scarab.sol

Locations

```
4
5   pragma solidity >=0.6.0 <0.8.0;
6
7   /*
8   * @dev Provides information about the current execution context, including the
9
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 33

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

- Scarab.sol

Locations

```
32
33   pragma solidity >=0.6.0 <0.8.0;
34
35   /**
36   * @dev Interface of the ERC20 standard as defined in the EIP.
37
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 114

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

- Scarab.sol

Locations

```
113
114   pragma solidity >=0.6.0 <0.8.0;
115
116   /**
117   * @dev Wrappers over Solidity's arithmetic operations with added overflow
118
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 332

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

- Scarab.sol

Locations

```
331
332   pragma solidity >=0.6.0 <0.8.0;
333
334
335
336
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 640

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

- Scarab.sol

Locations

```
639
640   pragma solidity >=0.6.0 <0.8.0;
641
642
643   /**
644
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 684

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

- Scarab.sol

Locations

```
683
684   pragma solidity >=0.6.0 <0.8.0;
685
686   /**
687   * @dev Standard math utilities missing in the Solidity language.
688
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 882

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

- Scarab.sol

Locations

```
881
882   pragma solidity >=0.6.0 <0.8.0;
883
884
885   // File @openzeppelin/contracts/access/Ownable.sol@v3.4.2
886
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 889

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

- Scarab.sol

Locations

```
888
889   pragma solidity >=0.6.0 <0.8.0;
890
891   /**
892   * @dev Contract module which provides a basic access control mechanism, where
893
```

SWC-110 | PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.

LINE 1041

low SEVERITY

The public state variable "taxTiersTwaps" in "Scarab" contract has type "uint256[]" and can cause an exception in case of use of invalid array index value.

Source File

- Scarab.sol

Locations

```
1040   // Tax Tiers
1041   uint256[] public taxTiersTwaps = [0, 5e17, 6e17, 7e17, 8e17, 9e17, 9.5e17, 1e18,
1.05e18, 1.10e18, 1.20e18, 1.30e18, 1.40e18, 1.50e18];
1042   uint256[] public taxTiersRates = [2000, 1900, 1800, 1700, 1600, 1500, 1500, 1500,
1500, 1400, 900, 400, 200, 100];
1043
1044   // Sender addresses excluded from Tax
1045
```

SWC-110 | PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.

LINE 1042

low SEVERITY

The public state variable "taxTiersRates" in "Scarab" contract has type "uint256[]" and can cause an exception in case of use of invalid array index value.

Source File

- Scarab.sol

Locations

```
1041   uint256[] public taxTiersTwaps = [0, 5e17, 6e17, 7e17, 8e17, 9e17, 9.5e17, 1e18,
1.05e18, 1.10e18, 1.20e18, 1.30e18, 1.40e18, 1.50e18];
1042   uint256[] public taxTiersRates = [2000, 1900, 1800, 1700, 1600, 1500, 1500, 1500,
1500, 1400, 900, 400, 200, 100];
1043
1044   // Sender addresses excluded from Tax
1045   mapping(address => bool) public excludedAddresses;
1046
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1092

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
1091   if (_index > 0) {
1092   require(_value > taxTiersTwaps[_index - 1]);
1093   }
1094   if (_index < getTaxTiersTwapsCount().sub(1)) {
1095   require(_value < taxTiersTwaps[_index + 1]);
1096
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1095

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
1094   if (_index < getTaxTiersTwapsCount().sub(1)) {
1095   require(_value < taxTiersTwaps[_index + 1]);
1096   }
1097   taxTiersTwaps[_index] = _value;
1098   return true;
1099
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1097

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
1096   }
1097   taxTiersTwaps[_index] = _value;
1098   return true;
1099   }
1100
1101
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1104

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
1103   require(_index < getTaxTiersRatesCount(), "Index has to lower than count of tax
tiers");
1104   taxTiersRates[_index] = _value;
1105   return true;
1106   }
1107
1108
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1123

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
1122   for (uint8 tierId = uint8(getTaxTiersTwapsCount()).sub(1); tierId >= 0; --tierId)
{
1123   if (_tombPrice >= taxTiersTwaps[tierId]) {
1124   require(taxTiersRates[tierId] < 10000, "tax equal or bigger to 100%");
1125   taxRate = taxTiersRates[tierId];
1126   return taxTiersRates[tierId];
1127
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1124

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
1123   if (_tombPrice >= taxTiersTwaps[tierId]) {
1124   require(taxTiersRates[tierId] < 10000, "tax equal or bigger to 100%");
1125   taxRate = taxTiersRates[tierId];
1126   return taxTiersRates[tierId];
1127   }
1128
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1125

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
1124   require(taxTiersRates[tierId] < 10000, "tax equal or bigger to 100%");
1125   taxRate = taxTiersRates[tierId];
1126   return taxTiersRates[tierId];
1127   }
1128   }
1129
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1126

low SEVERITY

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

Source File

- Scarab.sol

Locations

```
1125   taxRate = taxTiersRates[tierId];
1126   return taxTiersRates[tierId];
1127   }
1128   }
1129   }
1130
```

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.