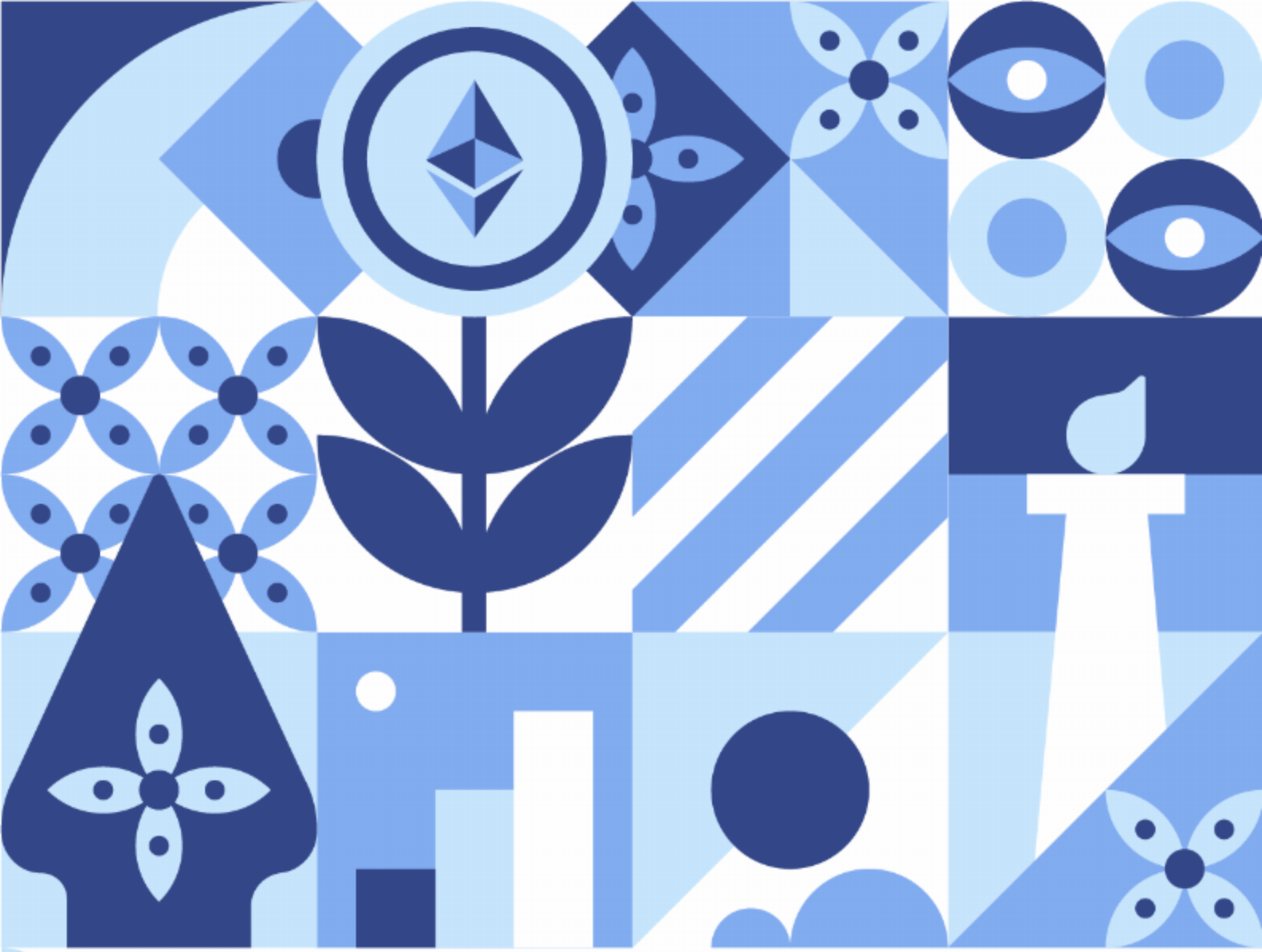

Shibarium DAO

Smart Contract Audit Report

SYSFIXED

30 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Shibarium DAO	SHIBDAO	Ethereum

Addresses

Contract address	0x15316d2438A8D7D534e4233B8E0edacD64c9FCde
Contract deployer address	0xd4960267891F83f53979Ce5a439849EfA81B7549

Project Website

https://shibariumdao.io/

Codebase

https://etherscan.io/address/0x15316d2438A8D7D534e4233B8E0edacD64c9FCde#code

SUMMARY

Shibarium DAO is the first decentralized organization founded to develop a community on the Shibarium blockchain. In addition to creating the community itself, we are also creating a special place for them in the form of a decentralized community platform where they can feel completely free.

Contract Summary

Documentation Quality

Shibarium DAO 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 Shibarium DAO with the discovery of several low issues.

Test Coverage

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

Audit Findings Summary

- SWC-100 SWC-108 | Explicitly define visibility for all state variables on lines 139, 140, 146, 147, 148, 149, 151, 152, 156, 157, 158, 160, 161, 162, 163, 164, 170, 171, 173, 174, 175, 177, 184, 185, 186, 190, 193 and 199.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 146, 146, 147, 147, 148, 148, 149, 149, 162, 191, 192, 253, 253, 257, 257, 261, 261, 305, 312, 312, 321, 321, 338, 339, 339, 349, 372, 403, 413, 434, 435, 443, 461, 467, 467, 497, 500, 501, 509, 509, 511, 513, 581, 581, 582, 582, 590, 590, 621, 622, 622, 626 and 626.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 21.
- 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 307, 307, 311, 312, 530 and 531.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 481, 485, 488 and 488.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 337, 424, 461, 472, 476, 481, 484, 485, 487, 488 and 497.

CONCLUSION

We have audited the Shibarium DAO project released on January 2023 to discover issues and identify potential security vulnerabilities in Shibarium DAO 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 Shibarium DAO smart contract code do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general but there are still much low risk issues that must be fixed. The low-risk issues found are some arithmetic operation issues, a floating pragma is set, a state variable visibility is not set, weak sources of randomness, tx.origin as a part of authorization control and out of bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value. We recommend solving with lock the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen. Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables. The "tx.origin" should not be used for authorization. Use "msg.sender" instead. Using a commitment scheme, e.g. RANDAO. Using external sources of randomness via oracles, e.g. Oraclize. Note that this approach requires trusting in oracle, thus it may be reasonable to use multiple oracles. Using Bitcoin block hashes, as they are more expensive to mine.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	ISSUE FOUND
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
Unprotected Ether Withdrawal	SWC-105	Due to missing or insufficient access controls, malicious parties can withdraw from the contract.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Uninitialized Storage Pointer	SWC-109	Uninitialized local storage variables can point to unexpected storage locations in the contract.	PASS
Assert Violation	SWC-110 SWC-123	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS

DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	ISSUE FOUND
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique ID	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	ISSUE FOUND
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	Sunday Jan 29 2023 21:27:07 GMT+0000 (Coordinated Universal Time)
Finished	Monday Jan 30 2023 02:38:28 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	ShibariumDAO.sol

Detected Issues

ID	Title	Severity	Status
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged

SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged

SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged

SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 146

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
145
146   uint256 _totalSupply = 1_000_000_000 * (10**_decimals);
147   uint256 _maxBuyTxAmount = (_totalSupply * 1) / 10;
148   uint256 _maxSellTxAmount = (_totalSupply * 1) / 10;
149   uint256 _maxWalletSize = (_totalSupply * 1) / 10;
150
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 146

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
145
146   uint256 _totalSupply = 1_000_000_000 * (10**_decimals);
147   uint256 _maxBuyTxAmount = (_totalSupply * 1) / 10;
148   uint256 _maxSellTxAmount = (_totalSupply * 1) / 10;
149   uint256 _maxWalletSize = (_totalSupply * 1) / 10;
150
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 147

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
146   uint256 _totalSupply = 1_000_000_000 * (10**_decimals);
147   uint256 _maxBuyTxAmount = (_totalSupply * 1) / 10;
148   uint256 _maxSellTxAmount = (_totalSupply * 1) / 10;
149   uint256 _maxWalletSize = (_totalSupply * 1) / 10;
150
151
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 147

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
146   uint256 _totalSupply = 1_000_000_000 * (10**_decimals);
147   uint256 _maxBuyTxAmount = (_totalSupply * 1) / 10;
148   uint256 _maxSellTxAmount = (_totalSupply * 1) / 10;
149   uint256 _maxWalletSize = (_totalSupply * 1) / 10;
150
151
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 148

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
147   uint256 _maxBuyTxAmount = (_totalSupply * 1) / 10;
148   uint256 _maxSellTxAmount = (_totalSupply * 1) / 10;
149   uint256 _maxWalletSize = (_totalSupply * 1) / 10;
150
151   mapping(address => uint256) _balances;
152
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 148

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
147   uint256 _maxBuyTxAmount = (_totalSupply * 1) / 10;
148   uint256 _maxSellTxAmount = (_totalSupply * 1) / 10;
149   uint256 _maxWalletSize = (_totalSupply * 1) / 10;
150
151   mapping(address => uint256) _balances;
152
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 149

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
148   uint256 _maxSellTxAmount = (_totalSupply * 1) / 10;
149   uint256 _maxWalletSize = (_totalSupply * 1) / 10;
150
151   mapping(address => uint256) _balances;
152   mapping(address => mapping(address => uint256)) _allowances;
153
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 149

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
148   uint256 _maxSellTxAmount = (_totalSupply * 1) / 10;
149   uint256 _maxWalletSize = (_totalSupply * 1) / 10;
150
151   mapping(address => uint256) _balances;
152   mapping(address => mapping(address => uint256)) _allowances;
153
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 162

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
161   uint256 liquidityFee = 300;
162   uint256 totalFee = marketingFee + liquidityFee;
163   uint256 sellBias = 0;
164   uint256 feeDenominator = 10000;
165
166
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 191

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
190   bool processEnabled = true;
191   uint256 public swapThreshold = _totalSupply / 1000;
192   uint256 public swapMinimum = _totalSupply / 10000;
193   bool inSwap;
194   modifier swapping() {
195
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 192

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
191   uint256 public swapThreshold = _totalSupply / 1000;
192   uint256 public swapMinimum = _totalSupply / 10000;
193   bool inSwap;
194   modifier swapping() {
195   inSwap = true;
196
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 253

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
252   function maxBuyTxTokens() external view returns (uint256) {
253   return _maxBuyTxAmount / (10**_decimals);
254   }
255
256   function maxSellTxTokens() external view returns (uint256) {
257
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 253

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
252   function maxBuyTxTokens() external view returns (uint256) {
253   return _maxBuyTxAmount / (10**_decimals);
254   }
255
256   function maxSellTxTokens() external view returns (uint256) {
257
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 257

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
256   function maxSellTxTokens() external view returns (uint256) {
257   return _maxSellTxAmount / (10**_decimals);
258   }
259
260   function maxWalletTokens() external view returns (uint256) {
261
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 257

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
256   function maxSellTxTokens() external view returns (uint256) {
257   return _maxSellTxAmount / (10**_decimals);
258   }
259
260   function maxWalletTokens() external view returns (uint256) {
261
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 261

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
260   function maxWalletTokens() external view returns (uint256) {
261   return _maxWalletSize / (10**_decimals);
262   }
263
264   function balanceOf(address account) public view override returns (uint256) {
265
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 261

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
260   function maxWalletTokens() external view returns (uint256) {
261   return _maxWalletSize / (10**_decimals);
262   }
263
264   function balanceOf(address account) public view override returns (uint256) {
265
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 305

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
304
305   for (uint256 i = 0; i < addresses.length; i++) {
306   if (
307   !liquidityPools[addresses[i]] && !liquidityCreator[addresses[i]]
308   ) {
309
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 312

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
311   addresses[i],
312   amounts[i] * (10**_decimals)
313   );
314   }
315   }
316
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 312

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
311   addresses[i],
312   amounts[i] * (10**_decimals)
313   );
314   }
315   }
316
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 321

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
320   uint256 amountETH = address(this).balance;
321   payable(devWallet).transfer((amountETH * amount) / 100);
322   }
323   }
324
325
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 321

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
320   uint256 amountETH = address(this).balance;
321   payable(devWallet).transfer((amountETH * amount) / 100);
322   }
323   }
324
325
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 338

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
337   launchedAt = block.number;
338   protectionTimer = block.timestamp + _protection;
339   protectionLimit = _limit * (10**_decimals);
340   }
341
342
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 339

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
338   protectionTimer = block.timestamp + _protection;
339   protectionLimit = _limit * (10**_decimals);
340   }
341
342   function enableProtection(bool _protect, uint256 _addTime)
343
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 339

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
338   protectionTimer = block.timestamp + _protection;
339   protectionLimit = _limit * (10**_decimals);
340   }
341
342   function enableProtection(bool _protect, uint256 _addTime)
343
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 349

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
348   require(_addTime < 1 days);
349   protectionTimer += _addTime;
350   }
351
352   function disableProtection() external onlyTeam {
353
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 372

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
371   _allowances[sender][msg.sender] =
372   _allowances[sender][msg.sender] -
373   amount;
374   }
375
376
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 403

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
402
403   _balances[sender] = _balances[sender] - amount;
404
405   uint256 amountReceived = feeExcluded(sender)
406   ? takeFee(recipient, amount)
407
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 413

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
412
413   _balances[recipient] = _balances[recipient] + amountReceived;
414
415   emit Transfer(sender, recipient, amountReceived);
416   return true;
417
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 434

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
433   ) internal returns (bool) {
434   _balances[sender] = _balances[sender] - amount;
435   _balances[recipient] = _balances[recipient] + amount;
436   emit Transfer(sender, recipient, amount);
437   return true;
438
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 435

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
434   _balances[sender] = _balances[sender] - amount;
435   _balances[recipient] = _balances[recipient] + amount;
436   emit Transfer(sender, recipient, amount);
437   return true;
438   }
439
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 443

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
442   require(
443   _balances[recipient] + amount <= walletLimit,
444   "Transfer amount exceeds the bag size."
445   );
446   }
447
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 461

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
460   isTxLimitExempt[sender] ||
461   lastBuy[recipient] + rateLimit <= block.number,
462   "Transfer rate limit exceeded."
463   );
464
465
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 467

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
466   require(
467   amount <= protectionLimit * (10**_decimals) &&
468   lastSell[sender] == 0 &&
469   protectionTimer > block.timestamp,
470   "Wallet protected, please contact support."
471
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 467

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
466   require(
467   amount <= protectionLimit * (10**_decimals) &&
468   lastSell[sender] == 0 &&
469   protectionTimer > block.timestamp,
470   "Wallet protected, please contact support."
471
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 497

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
496   function getTotalFee(bool selling) public view returns (uint256) {
497   if (launchedAt + deadBlocks >= block.number) {
498   return feeDenominator;
499   }
500   if (selling) return totalFee + sellBias;
501
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 500

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
499   }
500   if (selling) return totalFee + sellBias;
501   return totalFee - sellBias;
502   }
503
504
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 501

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
500   if (selling) return totalFee + sellBias;
501   return totalFee - sellBias;
502   }
503
504   function takeFee(address recipient, uint256 amount)
505
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 509

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
508   bool selling = liquidityPools[recipient];
509   uint256 feeAmount = (amount * getTotalFee(selling)) / feeDenominator;
510
511   _balances[address(this)] += feeAmount;
512
513
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 509

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
508   bool selling = liquidityPools[recipient];
509   uint256 feeAmount = (amount * getTotalFee(selling)) / feeDenominator;
510
511   _balances[address(this)] += feeAmount;
512
513
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 511

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
510
511   _balances[address(this)] += feeAmount;
512
513   return amount - feeAmount;
514   }
515
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 513

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
512
513   return amount - feeAmount;
514   }
515
516   function shouldSwapBack(address recipient) internal view returns (bool) {
517
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 581

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
580   );
581   _maxBuyTxAmount = (_totalSupply * buyNumerator) / divisor;
582   _maxSellTxAmount = (_totalSupply * sellNumerator) / divisor;
583   }
584
585
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 581

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
580   );
581   _maxBuyTxAmount = (_totalSupply * buyNumerator) / divisor;
582   _maxSellTxAmount = (_totalSupply * sellNumerator) / divisor;
583   }
584
585
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 582

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
581   _maxBuyTxAmount = (_totalSupply * buyNumerator) / divisor;
582   _maxSellTxAmount = (_totalSupply * sellNumerator) / divisor;
583   }
584
585   function setMaxWallet(uint256 numerator, uint256 divisor)
586
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 582

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
581   _maxBuyTxAmount = (_totalSupply * buyNumerator) / divisor;
582   _maxSellTxAmount = (_totalSupply * sellNumerator) / divisor;
583   }
584
585   function setMaxWallet(uint256 numerator, uint256 divisor)
586
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 590

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
589   require(numerator > 0 && divisor > 0 && divisor <= 10000);
590   _maxWalletSize = (_totalSupply * numerator) / divisor;
591   }
592
593   function setIsFeeExempt(address holder, bool exempt) external onlyOwner {
594
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 590

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
589   require(numerator > 0 && divisor > 0 && divisor <= 10000);
590   _maxWalletSize = (_totalSupply * numerator) / divisor;
591   }
592
593   function setIsFeeExempt(address holder, bool exempt) external onlyOwner {
594
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 621

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
620   processEnabled = _processEnabled;
621   swapThreshold = _totalSupply / _denominator;
622   swapMinimum = _swapMinimum * (10**_decimals);
623   }
624
625
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 622

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
621   swapThreshold = _totalSupply / _denominator;
622   swapMinimum = _swapMinimum * (10**_decimals);
623   }
624
625   function getCurrentSupply() public view returns (uint256) {
626
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 622

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
621   swapThreshold = _totalSupply / _denominator;
622   swapMinimum = _swapMinimum * (10**_decimals);
623   }
624
625   function getCurrentSupply() public view returns (uint256) {
626
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 626

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
625   function getCurrentSupply() public view returns (uint256) {
626   return _totalSupply - (balanceOf(DEAD) + balanceOf(ZERO));
627   }
628
629   event FundsDistributed(uint256 marketingFee);
630
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 626

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
625   function getCurrentSupply() public view returns (uint256) {
626   return _totalSupply - (balanceOf(DEAD) + balanceOf(ZERO));
627   }
628
629   event FundsDistributed(uint256 marketingFee);
630
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 21

low SEVERITY

The current pragma Solidity directive is ""^0.8.7"". 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

- ShibariumDAO.sol

Locations

```
20
21  pragma solidity ^0.8.7;
22
23  abstract contract Context {
24  function _msgSender() internal view returns (address payable) {
25
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 139

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
138   contract ShibariumDAO is IERC20, Ownable {
139   address DEAD = 0x000000000000000000000000000000000000dEaD;
140   address ZERO = 0x0000000000000000000000000000000000000000;
141
142   string constant _name = "Shibarium DAO";
143
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 140

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
139   address DEAD = 0x000000000000000000000000000000000000dEaD;
140   address ZERO = 0x0000000000000000000000000000000000000000;
141
142   string constant _name = "Shibarium DAO";
143   string constant _symbol = "SHIBDAO";
144
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 146

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
145
146   uint256 _totalSupply = 1_000_000_000 * (10**_decimals);
147   uint256 _maxBuyTxAmount = (_totalSupply * 1) / 10;
148   uint256 _maxSellTxAmount = (_totalSupply * 1) / 10;
149   uint256 _maxWalletSize = (_totalSupply * 1) / 10;
150
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 147

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
146   uint256 _totalSupply = 1_000_000_000 * (10**_decimals);
147   uint256 _maxBuyTxAmount = (_totalSupply * 1) / 10;
148   uint256 _maxSellTxAmount = (_totalSupply * 1) / 10;
149   uint256 _maxWalletSize = (_totalSupply * 1) / 10;
150
151
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 148

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
147   uint256 _maxBuyTxAmount = (_totalSupply * 1) / 10;
148   uint256 _maxSellTxAmount = (_totalSupply * 1) / 10;
149   uint256 _maxWalletSize = (_totalSupply * 1) / 10;
150
151   mapping(address => uint256) _balances;
152
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 149

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
148   uint256 _maxSellTxAmount = (_totalSupply * 1) / 10;
149   uint256 _maxWalletSize = (_totalSupply * 1) / 10;
150
151   mapping(address => uint256) _balances;
152   mapping(address => mapping(address => uint256)) _allowances;
153
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 151

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
150
151   mapping(address => uint256) _balances;
152   mapping(address => mapping(address => uint256)) _allowances;
153   mapping(address => uint256) public lastSell;
154   mapping(address => uint256) public lastBuy;
155
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 152

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
151   mapping(address => uint256) _balances;
152   mapping(address => mapping(address => uint256)) _allowances;
153   mapping(address => uint256) public lastSell;
154   mapping(address => uint256) public lastBuy;
155
156
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 156

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
155
156   mapping(address => bool) isFeeExempt;
157   mapping(address => bool) isTxLimitExempt;
158   mapping(address => bool) liquidityCreator;
159
160
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 157

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
156   mapping(address => bool) isFeeExempt;
157   mapping(address => bool) isTxLimitExempt;
158   mapping(address => bool) liquidityCreator;
159
160   uint256 marketingFee = 200;
161
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 158

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
157   mapping(address => bool) isTxLimitExempt;
158   mapping(address => bool) liquidityCreator;
159
160   uint256 marketingFee = 200;
161   uint256 liquidityFee = 300;
162
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 160

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
159
160   uint256 marketingFee = 200;
161   uint256 liquidityFee = 300;
162   uint256 totalFee = marketingFee + liquidityFee;
163   uint256 sellBias = 0;
164
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 161

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
160   uint256 marketingFee = 200;
161   uint256 liquidityFee = 300;
162   uint256 totalFee = marketingFee + liquidityFee;
163   uint256 sellBias = 0;
164   uint256 feeDenominator = 10000;
165
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 162

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
161   uint256 liquidityFee = 300;
162   uint256 totalFee = marketingFee + liquidityFee;
163   uint256 sellBias = 0;
164   uint256 feeDenominator = 10000;
165
166
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 163

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
162   uint256 totalFee = marketingFee + liquidityFee;
163   uint256 sellBias = 0;
164   uint256 feeDenominator = 10000;
165
166   address payable public liquidityFeeReceiver = payable(address(this));
167
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 164

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
163   uint256 sellBias = 0;
164   uint256 feeDenominator = 10000;
165
166   address payable public liquidityFeeReceiver = payable(address(this));
167   address public marketingFeeReceiver;
168
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 170

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
169   IDEXRouter public router;
170   address routerAddress = 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D;
171   mapping(address => bool) liquidityPools;
172   mapping(address => uint256) public protected;
173   bool protectionEnabled = true;
174
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 171

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
170   address routerAddress = 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D;
171   mapping(address => bool) liquidityPools;
172   mapping(address => uint256) public protected;
173   bool protectionEnabled = true;
174   bool protectionDisabled = false;
175
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 173

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
172   mapping(address => uint256) public protected;
173   bool protectionEnabled = true;
174   bool protectionDisabled = false;
175   uint256 protectionLimit;
176   uint256 public protectionCount;
177
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 174

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
173   bool protectionEnabled = true;
174   bool protectionDisabled = false;
175   uint256 protectionLimit;
176   uint256 public protectionCount;
177   uint256 protectionTimer;
178
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 175

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
174   bool protectionDisabled = false;
175   uint256 protectionLimit;
176   uint256 public protectionCount;
177   uint256 protectionTimer;
178
179
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 177

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
176   uint256 public protectionCount;
177   uint256 protectionTimer;
178
179   address public pair;
180
181
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 184

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
183   uint256 public deadBlocks;
184   bool startBullRun = false;
185   bool pauseDisabled = false;
186   bool _feeOn = true;
187   uint256 public rateLimit = 2;
188
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 185

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
184   bool startBullRun = false;
185   bool pauseDisabled = false;
186   bool _feeOn = true;
187   uint256 public rateLimit = 2;
188
189
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 186

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
185   bool pauseDisabled = false;
186   bool _feeOn = true;
187   uint256 public rateLimit = 2;
188
189   bool public swapEnabled = false;
190
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 190

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
189   bool public swapEnabled = false;
190   bool processEnabled = true;
191   uint256 public swapThreshold = _totalSupply / 1000;
192   uint256 public swapMinimum = _totalSupply / 10000;
193   bool inSwap;
194
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 193

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
192   uint256 public swapMinimum = _totalSupply / 10000;
193   bool inSwap;
194   modifier swapping() {
195   inSwap = true;
196   _;
197
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 199

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
198   }
199   address devWallet;
200   modifier onlyTeam() {
201   require(_msgSender() == devWallet, "Caller is not a team member");
202   _;
203
```

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

LINE 481

low SEVERITY

The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead.

Source File

- ShibariumDAO.sol

Locations

```
480   protectionTimer > block.timestamp &&
481   lastBuy[tx.origin] == block.number &&
482   protected[recipient] == 0
483   ) {
484   protected[recipient] = block.number;
485
```

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

LINE 485

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
484   protected[recipient] = block.number;
485   emit ProtectedWallet(tx.origin, recipient, block.number, 1);
486   }
487   lastBuy[recipient] = block.number;
488   if (tx.origin != recipient) lastBuy[tx.origin] = block.number;
489
```

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

LINE 488

low SEVERITY

The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead.

Source File

- ShibariumDAO.sol

Locations

```
487   lastBuy[recipient] = block.number;
488   if (tx.origin != recipient) lastBuy[tx.origin] = block.number;
489   }
490   }
491
492
```

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

LINE 488

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
487   lastBuy[recipient] = block.number;
488   if (tx.origin != recipient) lastBuy[tx.origin] = block.number;
489   }
490   }
491
492
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 307

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
306   if (
307   !liquidityPools[addresses[i]] && !liquidityCreator[addresses[i]]
308   ) {
309   _basicTransfer(
310   from,
311
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 307

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
306   if (
307   !liquidityPools[addresses[i]] && !liquidityCreator[addresses[i]]
308   ) {
309   _basicTransfer(
310   from,
311
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 311

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
310   from,
311   addresses[i],
312   amounts[i] * (10**_decimals)
313   );
314   }
315
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 312

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
311   addresses[i],
312   amounts[i] * (10**_decimals)
313   );
314   }
315   }
316
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 530

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
529   address[] memory path = new address[](2);
530   path[0] = address(this);
531   path[1] = router.WETH();
532
533   router.swapExactTokensForETHSupportingFeeOnTransferTokens(
534
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 531

low SEVERITY

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

Source File

- ShibariumDAO.sol

Locations

```
530   path[0] = address(this);
531   path[1] = router.WETH();
532
533   router.swapExactTokensForETHSupportingFeeOnTransferTokens(
534   amountToSwap,
535
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 337

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- ShibariumDAO.sol

Locations

```
336   startBullRun = true;
337   launchedAt = block.number;
338   protectionTimer = block.timestamp + _protection;
339   protectionLimit = _limit * (10**_decimals);
340   }
341
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 424

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- ShibariumDAO.sol

Locations

```
423   function launch() internal {
424   launchedAt = block.number;
425   launchedTime = block.timestamp;
426   swapEnabled = true;
427   }
428
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 461

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- ShibariumDAO.sol

Locations

```
460   isTxLimitExempt[sender] ||
461   lastBuy[recipient] + rateLimit <= block.number,
462   "Transfer rate limit exceeded."
463   );
464
465
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 472

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- ShibariumDAO.sol

Locations

```
471   );
472   lastSell[sender] = block.number;
473   }
474
475   if (liquidityPools[recipient]) {
476
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 476

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- ShibariumDAO.sol

Locations

```
475   if (liquidityPools[recipient]) {
476   lastSell[sender] = block.number;
477   } else if (feeExcluded(sender)) {
478   if (
479   protectionEnabled &&
480
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 481

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- ShibariumDAO.sol

Locations

```
480   protectionTimer > block.timestamp &&
481   lastBuy[tx.origin] == block.number &&
482   protected[recipient] == 0
483   ) {
484   protected[recipient] = block.number;
485
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 484

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- ShibariumDAO.sol

Locations

```
483   ) {
484   protected[recipient] = block.number;
485   emit ProtectedWallet(tx.origin, recipient, block.number, 1);
486   }
487   lastBuy[recipient] = block.number;
488
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 485

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- ShibariumDAO.sol

Locations

```
484   protected[recipient] = block.number;
485   emit ProtectedWallet(tx.origin, recipient, block.number, 1);
486   }
487   lastBuy[recipient] = block.number;
488   if (tx.origin != recipient) lastBuy[tx.origin] = block.number;
489
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 487

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- ShibariumDAO.sol

Locations

```
486   }
487   lastBuy[recipient] = block.number;
488   if (tx.origin != recipient) lastBuy[tx.origin] = block.number;
489   }
490   }
491
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 488

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- ShibariumDAO.sol

Locations

```
487   lastBuy[recipient] = block.number;
488   if (tx.origin != recipient) lastBuy[tx.origin] = block.number;
489   }
490   }
491
492
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 497

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- ShibariumDAO.sol

Locations

```
496   function getTotalFee(bool selling) public view returns (uint256) {
497   if (launchedAt + deadBlocks >= block.number) {
498   return feeDenominator;
499   }
500   if (selling) return totalFee + sellBias;
501
```

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.