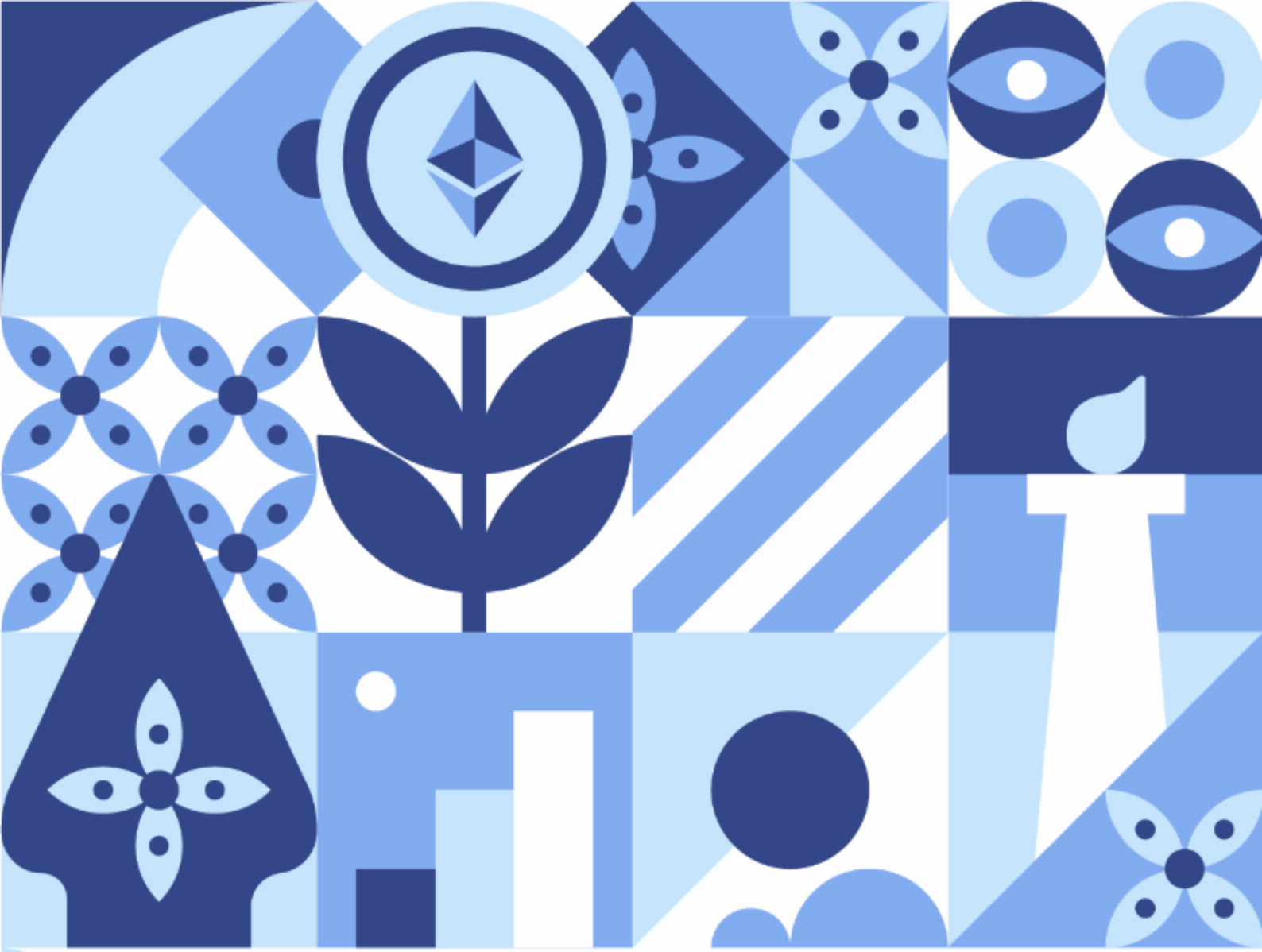

Alcazar

Smart Contract Audit Report

SYSFIXED

25 Oct 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Alcazar	ALCAZAR	Ethereum

Addresses

Contract address	0x10f44a834097469ac340592d28c479c442e99bfe
Contract deployer address	0x2FF9d7be466f674c8640466A55fFdd02b3a00864

Project Website

https://alcazar.world/

Codebase

https://etherscan.io/address/0x10f44a834097469ac340592d28c479c442e99bfe#code

SUMMARY

Alcazar is building a next generation dApp with the purpose of making it possible for Alcazar to raffle away many high quality luxurious items in between holders. The raffles consist of items such as high-end luxury watches, vehicles, tickets to different events and high amounts of other tokens.

Contract Summary

Documentation Quality

Alcazar 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 Alcazar 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 978, 982 and 990.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 250, 282, 305, 306, 341, 377, 947, 947, 948, 948, 951, 951, 952, 952, 1013, 1014, 1017, 1147, 1154, 1162 and 1236.
- 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 1237, 1238, 1356, 1357, 1386 and 1387.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 1118.

CONCLUSION

We have audited the Alcazar project released on October 2022 to discover issues and identify potential security vulnerabilities in Alcazar Project. This process is used to find technical issues and security loopholes which might be found in the smart contract.

The security audit report provides satisfactory results with low-risk issues.

The issues found in the Alcazar 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, State variable visibility is not set, Potential use of "block.number" as a source of randomness, 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.	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.	PASS
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.	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	Monday Oct 24 2022 04:10:16 GMT+0000 (Coordinated Universal Time)
Finished	Tuesday Oct 25 2022 12:34:50 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Alcazar.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-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-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-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 250

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
249   function add(uint256 a, uint256 b) internal pure returns (uint256) {
250   uint256 c = a + b;
251   require(c >= a, "SafeMath: addition overflow");
252
253   return c;
254
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 282

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
281   require(b <= a, errorMessage);
282   uint256 c = a - b;
283
284   return c;
285   }
286
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 305

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
304
305   uint256 c = a * b;
306   require(c / a == b, "SafeMath: multiplication overflow");
307
308   return c;
309
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 306

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
305   uint256 c = a * b;
306   require(c / a == b, "SafeMath: multiplication overflow");
307
308   return c;
309   }
310
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 341

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
340   require(b > 0, errorMessage);
341   uint256 c = a / b;
342   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
343
344   return c;
345
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 377

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
376   require(b != 0, errorMessage);
377   return a % b;
378   }
379   }
380
381
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 947

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
946   // Supply
947   uint256 private _totalSupply = 1 * 1e9 * 1e18;
948   uint256 private minimumTokensBeforeSwap = _totalSupply * 25 / 100000;
949
950   // Restrictions
951
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 947

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
946   // Supply
947   uint256 private _totalSupply = 1 * 1e9 * 1e18;
948   uint256 private minimumTokensBeforeSwap = _totalSupply * 25 / 100000;
949
950   // Restrictions
951
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 948

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
947   uint256 private _totalSupply = 1 * 1e9 * 1e18;
948   uint256 private minimumTokensBeforeSwap = _totalSupply * 25 / 100000;
949
950   // Restrictions
951   uint256 public _maxTxAmount = (_totalSupply * 8) / 1000;
952
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 948

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
947   uint256 private _totalSupply = 1 * 1e9 * 1e18;
948   uint256 private minimumTokensBeforeSwap = _totalSupply * 25 / 100000;
949
950   // Restrictions
951   uint256 public _maxTxAmount = (_totalSupply * 8) / 1000;
952
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 951

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
950   // Restrictions
951   uint256 public _maxTxAmount = (_totalSupply * 8) / 1000;
952   uint256 public _walletMax = (_totalSupply * 8) / 1000;
953   bool public checkWalletLimit = true;
954
955
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 951

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
950   // Restrictions
951   uint256 public _maxTxAmount = (_totalSupply * 8) / 1000;
952   uint256 public _walletMax = (_totalSupply * 8) / 1000;
953   bool public checkWalletLimit = true;
954
955
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 952

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
951   uint256 public _maxTxAmount = (_totalSupply * 8) / 1000;
952   uint256 public _walletMax = (_totalSupply * 8) / 1000;
953   bool public checkWalletLimit = true;
954
955   // wallets
956
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 952

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
951   uint256 public _maxTxAmount = (_totalSupply * 8) / 1000;
952   uint256 public _walletMax = (_totalSupply * 8) / 1000;
953   bool public checkWalletLimit = true;
954
955   // wallets
956
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1013

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
1012   // load total fees
1013   totalFeesBuy = operationsFeeBuy + liquidityFeeBuy;
1014   totalFeesSell = operationsFeeSell + liquidityFeeSell;
1015
1016   // load total distribution
1017
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1014

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
1013   totalFeesBuy = operationsFeeBuy + liquidityFeeBuy;
1014   totalFeesSell = operationsFeeSell + liquidityFeeSell;
1015
1016   // load total distribution
1017   _totalDistributionShares = _liquiditySharePercentage + _operationsSharePercentage;
1018
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1017

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
1016   // load total distribution
1017   _totalDistributionShares = _liquiditySharePercentage + _operationsSharePercentage;
1018
1019   // create router ------------------------------
1020   IUniswapV2Router02 _uniswapV2Router;
1021
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1147

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
1146   liquidityFeeBuy = _liquidityFee;
1147   totalFeesBuy = operationsFeeBuy + liquidityFeeBuy;
1148   require(totalFeesBuy <= maxTotalFeeBuy, "Must keep fees at maxTotalFeeBuy or
less");
1149   }
1150
1151
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1154

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
1153   liquidityFeeSell = _liquidityFee;
1154   totalFeesSell = operationsFeeSell + liquidityFeeSell;
1155   require(totalFeesSell <= maxTotalFeeSell, "Must keep fees at maxTotalFeeSell or
less");
1156   }
1157
1158
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1162

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
1161   _operationsSharePercentage = newOperationsShare;
1162   _totalDistributionShares = _liquiditySharePercentage + _operationsSharePercentage;
1163   require(_totalDistributionShares == 100, "Distribution needs to total to 100");
1164   }
1165
1166
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1236

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
1235   require(airdropWallets.length == amount.length, "airdropToWallets:: Arrays must be
the same length");
1236   for(uint256 i = 0; i < airdropWallets.length; i++){
1237   address wallet = airdropWallets[i];
1238   uint256 airdropAmount = amount[i];
1239   emit Transfer(msg.sender, wallet, airdropAmount);
1240
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 978

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

- Alcazar.sol

Locations

```
977   // max amounts
978   mapping (address => uint256) _balances;
979   mapping (address => mapping (address => uint256)) private _allowances;
980   mapping (address => bool) public isExcludedFromFee;
981   mapping (address => bool) public isWalletLimitExempt;
982
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 982

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

- Alcazar.sol

Locations

```
981   mapping (address => bool) public isWalletLimitExempt;
982   mapping (address => bool) isTxLimitExempt;
983
984   // Router Information
985   mapping (address => bool) public isMarketPair;
986
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 990

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
989   // toggle swap back (fees)
990   bool inSwapAndLiquify;
991   uint256 public tokensForLiquidity;
992   uint256 public tokensForOperations;
993
994
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1237

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
1236   for(uint256 i = 0; i < airdropWallets.length; i++){
1237   address wallet = airdropWallets[i];
1238   uint256 airdropAmount = amount[i];
1239   emit Transfer(msg.sender, wallet, airdropAmount);
1240   }
1241
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1238

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
1237   address wallet = airdropWallets[i];
1238   uint256 airdropAmount = amount[i];
1239   emit Transfer(msg.sender, wallet, airdropAmount);
1240   }
1241   }
1242
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1356

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
1355   address[] memory path = new address[](2);
1356   path[0] = address(this);
1357   path[1] = uniswapV2Router.WETH();
1358   _approve(address(this), address(uniswapV2Router), tokenAmount);
1359   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
1360
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1357

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
1356   path[0] = address(this);
1357   path[1] = uniswapV2Router.WETH();
1358   _approve(address(this), address(uniswapV2Router), tokenAmount);
1359   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
1360   tokenAmount,
1361
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1386

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
1385   address[] memory path = new address[](2);
1386   path[0] = uniswapV2Router.WETH();
1387   path[1] = address(this);
1388
1389   // make the swap
1390
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1387

low SEVERITY

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

Source File

- Alcazar.sol

Locations

```
1386   path[0] = uniswapV2Router.WETH();
1387   path[1] = address(this);
1388
1389   // make the swap
1390   uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
ETHAmountInWei}(
1391
```

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

LINE 1118

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

- Alcazar.sol

Locations

```
1117   function getBlock()public view returns (uint256) {
1118   return block.number;
1119   }
1120
1121   // @dev Owner functions start ------------------------------------
1122
```

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.