AGRITECH

Smart Contract Audit Report

SYSFIXED

22 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
AGRITECH	AGT	Binance Smart Chain

Addresses

Contract address	0x8B6742CB878f3E8b5E58424F4fDeeAfF1dFF9D85
Contract deployer address	0x11d440e6ee2684A4c28a415383745bEC35D7cE38

Project Website

https://www.agri-tech.io/

Codebase

https://bscscan.com/address/0x8B6742CB878f3E8b5E58424F4fDeeAfF1dFF9D85#code

SUMMARY

AGRITECH is the first peer-to-peer marketplace for Agriculture utilizing Blockchain and Web3. Agritech utilized Thailand's agriculture to launch proprietary Traceability and AI technology. AGRITECH set aside 122k Tokens for DEX (PancakeSwap) & 60mil Tokens for CEX (CoinW) Partnership w/Thailand, Haiti and over 30+ partners.

Contract Summary

Documentation Quality

AGRITECH 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 AGRITECH 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 105 and 126.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 117, 249, 270, 364, 365, 366, 376 and 377.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 6.
- SWC-110 | It is recommended to use revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 365 and 366.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 298.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 352.

CONCLUSION

We have audited the AGRITECH project released on January 2023 to discover issues and identify potential security vulnerabilities in AGRITECH Project. This process is used to find technical issues and security loopholes which might be found in the smart contract.

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

The issues found in the AGRITECH 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-level issues found are some arithmetic operation issues, a floating pragma is set, a state variable visibility is not set, weak sources of randomness and out of bounds array access which the index access expression can cause an exception in case of 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.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	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
Assert Violation	SWC-110	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 Caller	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
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
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

SMART CONTRACT ANALYSIS

Started	Saturday Jan 21 2023 22:52:50 GMT+0000 (Coordinated Universal Time)
Finished	Sunday Jan 22 2023 08:57:26 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	AGRITECH.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-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-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-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 117

low SEVERITY

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

Source File

- AGRITECH.sol

Locations

```
116   uint8 constant private _decimals = 18;
117   uint256 constant private _tTotal = startingSupply * 10**_decimals;
118
119   bool public taxesAreLocked;
120   IRouter02 public dexRouter;
121
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 249

low SEVERITY

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

Source File

- AGRITECH.sol

Locations

```
248   if (_allowances[sender][msg.sender] != type(uint256).max) {
249   _allowances[sender][msg.sender] -= amount;
250   }
251
252   return _transfer(sender, recipient, amount);
253
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 270

low SEVERITY

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

Source File

- AGRITECH.sol

Locations

```
269   function getCirculatingSupply() public view returns (uint256) {
270   return (_tTotal - (balanceOf(DEAD) + balanceOf(address(0))));
271   }
272
273   function removeSniper(address account) external onlyOwner {
274
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 364

low SEVERITY

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

Source File

- AGRITECH.sol

Locations

```
363   require(accounts.length == amounts.length, "Lengths do not match.");
364   for (uint16 i = 0; i < accounts.length; i++) {
365   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
366   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, true);
367   }
368
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 365

low SEVERITY

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

Source File

- AGRITECH.sol

Locations

```
364   for (uint16 i = 0; i < accounts.length; i++) {
365   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
366   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, true);
367   }
368   }
369
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 366

low SEVERITY

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

Source File

- AGRITECH.sol

Locations

```
365   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
366   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, true);
367   }
368   }
369
370
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 376

low SEVERITY

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

Source File

- AGRITECH.sol

Locations

```
375   }
376   _tOwned[from] -= amount;
377   _tOwned[to] += amount;
378   emit Transfer(from, to, amount);
379   if (!_hasLiqBeenAdded) {
380
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 377

low SEVERITY

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

Source File

- AGRITECH.sol

Locations

```
376   _tOwned[from] -= amount;
377   _tOwned[to] += amount;
378   emit Transfer(from, to, amount);
379   if (!_hasLiqBeenAdded) {
380   _checkLiquidityAdd(from, to);
381
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 6

low SEVERITY

The current pragma Solidity directive is ""=0.6.0<0.9.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- AGRITECH.sol

Locations

```
5   // SPDX-License-Identifier: MIT
6   pragma solidity >=0.6.0 <0.9.0;
7
8   interface IERC20 {
9   function totalSupply() external view returns (uint256);
10
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 105

low SEVERITY

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

Source File

- AGRITECH.sol

Locations

```
104   mapping (address => uint256) private _tOwned;
105   mapping (address => bool) lpPairs;
106   uint256 private timeSinceLastPair = 0;
107   mapping (address => mapping (address => uint256)) private _allowances;
108   mapping (address => bool) private _liquidityHolders;
109
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 126

low SEVERITY

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

Source File

- AGRITECH.sol

Locations

```
125   bool public _hasLiqBeenAdded = false;
126   Protections protections;
127
128   constructor () payable {
129   // Set the owner.
130
```

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

LINE 298

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

- AGRITECH.sol

Locations

```
297   && to != _owner
298   && tx.origin != _owner
299   && !_liquidityHolders[to]
300   && !_liquidityHolders[from]
301   && to != DEAD
302
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 365

low SEVERITY

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

Source File

- AGRITECH.sol

Locations

```
364   for (uint16 i = 0; i < accounts.length; i++) {
365   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
366   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, true);
367   }
368   }
369
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 366

low SEVERITY

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

Source File

- AGRITECH.sol

Locations

```
365   require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
366   finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, true);
367   }
368   }
369
370
```

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

LINE 352

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

- AGRITECH.sol

Locations

```
351   }
352   try protections.setLaunch(lpPair, uint32(block.number), uint64(block.timestamp),
_decimals) {} catch {}
353   tradingEnabled = true;
354   allowedPresaleExclusion = false;
355   }
356
```

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.