AudiTBlock

GOCHAN

v0.8.18+commit.c7e474f2

Low-Risk	Medium-Risk	High-Risk
low-risk code	medium-risk code	high-risk code

GOCHAN

Contract Deployed On etherscan.io

0x92f702A11cd98cF05CF5947C39ff5B4cE06c2099

Disclaimer

AudiTBlock is not responsible if a project turns out to be a scam, rug-pull or honeypot. We only provide a detailed analysis for your own research.

AudiTBlock is not responsible for any financial losses. Nothing in this contract audit is financial advice, please do your ownresearch.

The information provided in this audit is for informational purposes only and should not be considered investment advice. We does not endorse, recommend, support or suggest to invest in any project.

AudiTBlock can not be held responsible for when a project turns out to be a rug-pull, honeypot or scam.

Tokenomics

- Etherscan.io

Source Code

- AudiTBlock was complete audit phases to perform an audit based on the following smart contract:

- https://etherscan.io/token/0x92f702a11cd98cf05cf5947c39ff5b4ce06c2099#code

Snapshot 0.1

```
Context is re-used:
	- Context (node_modules/@openzeppelin/contracts/utils/Context.sol#16-24)
	- Context (contracts/openzeppelin-contracts/contracts/utils/Context.sol#16-24)
ERC20 is re-used:
	- ERC20 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#38-365)
	- ERC20 (contracts/openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#35-389)
IERC20Metadata is re-used:
	- IERC20Metadata (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#13-28)
	- IERC20Metadata (contracts/openzeppelin-contracts/contracts/token/ERC20/extensions/IERC20Metadata.sol#13-28)
IERC20 is re-used:
	- IERC20 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#9-78)
	- IERC20 (contracts/openzeppelin-contracts/contracts/token/ERC20/IERC20.sol#9-82)
Ownable is re-used:
	- Ownable (node_modules/@openzeppelin/contracts/access/Ownable.sol#20-83)
	- Ownable (contracts/openzeppelin-contracts/contracts/access/Ownable.sol#20-83)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#name-reused


Reentrancy in Gochan.addInitialLiquidity(uint256,uint256) (contracts/Gochan.sol#73-100):
	External calls:
	- pair = IUniswapV2Factory(router.factory()).createPair(address(this),router.WETH()) (contracts/Gochan.sol#83-86)
	- router.addLiquidityETH{value: ethAmount}(address(this),liquidityAmount,0,0,msg.sender,block.timestamp + 900) (contracts/Gochan.sol#89-96)
	External calls sending eth:
	- router.addLiquidityETH{value: ethAmount}(address(this),liquidityAmount,0,0,msg.sender,block.timestamp + 900) (contracts/Gochan.sol#89-96)
	State variables written after the call(s):
	- blockStart = block.number (contracts/Gochan.sol#98)
	Gochan.blockStart (contracts/Gochan.sol#14) can be used in cross function reentrancies:
	- Gochan._transfer(address,address,uint256) (contracts/Gochan.sol#102-172)
	- Gochan.addInitialLiquidity(uint256,uint256) (contracts/Gochan.sol#73-100)
	- Gochan.blockStart (contracts/Gochan.sol#14)
	- timeStart = block.timestamp (contracts/Gochan.sol#99)
	Gochan.timeStart (contracts/Gochan.sol#15) can be used in cross function reentrancies:
	- Gochan._transfer(address,address,uint256) (contracts/Gochan.sol#102-172)
	- Gochan.addInitialLiquidity(uint256,uint256) (contracts/Gochan.sol#73-100)
	- Gochan.timeStart (contracts/Gochan.sol#15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
```

Snapshot 0.1

Gochan._transfer(address,address,uint256) (contracts/Gochan.sol#102-172) uses a dangerous strict equality:
- blockStart == 0 || timeStart == 0 (contracts/Gochan.sol#109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in Gochan._transfer(address,address,uint256) (contracts/Gochan.sol#102-172):
External calls:
- _swapTokensToEth(toSwap,recipient) (contracts/Gochan.sol#168)
- router.swapExactTokensForETH(tokenAmount,0,path,recip,block.timestamp + 900) (contracts/Gochan.sol#193-199)
State variables written after the call(s):
- super._transfer(from,to,toRecip) (contracts/Gochan.sol#171)
- _balances[from] = fromBalance - amount (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#231)
- _balances[to] += amount (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#234)
ERC20._balances (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#39) can be used in cross function reentrancies:
- ERC20._mint(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#251-264)
- ERC20._transfer(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#222-240)
- ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#101-103)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Gochan.addInitialLiquidity(uint256,uint256) (contracts/Gochan.sol#73-100) ignores return value by router.addLiquidityETH{value: ethAmount}(address(this),liquidityAmount,0,0,msg.sender,block.timestamp + 900) (contracts/Gochan.sol#89-96)
Gochan._swapTokensToEth(uint256,address) (contracts/Gochan.sol#185-200) ignores return value by router.swapExactTokensForETH(tokenAmount,0,path,recip,block.timestamp + 900) (contracts/Gochan.sol#193-199)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Tested Contract Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise,
after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed
condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5
Contracts/Gochan.sol	93b93518d9237edd5803dcf07ef3928b

Used Code from other Frameworks/Smart Contracts (direct imports)

Dependency / Import Path	Source Sha1 Hash
Contracts/openzeppelin, v2-core, v2-periphery	a12ab53ac8ccc1d75572fbffa51284d58517d643

Snapshot 0.2

Function IUniswapV2Router01.WETH() (contracts/v2-periphery/contracts/interfaces/IUniswapV2Router01.sol#5) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/v2-periphery/contracts/interfaces/IUniswapV2Router01.sol#10) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/v2-periphery/contracts/interfaces/IUniswapV2Router01.sol#11)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

ERC20._name (contracts/openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#42) should be immutable
ERC20._symbol (contracts/openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#43) should be immutable
Gochan.disabledBlocks (contracts/Gochan.sol#13) should be immutable
Gochan.fxEnd (contracts/Gochan.sol#25) should be immutable
Gochan.maxTx (contracts/Gochan.sol#23) should be immutable
Gochan.maxWallet (contracts/Gochan.sol#22) should be immutable
Gochan.recipient (contracts/Gochan.sol#19) should be immutable
Gochan.routerAddress (contracts/Gochan.sol#17) should be immutable
Gochan.tax (contracts/Gochan.sol#18) should be immutable
Gochan.taxThreshold (contracts/Gochan.sol#29) should be immutable
Gochan.txCooldown (contracts/Gochan.sol#24) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable

Different versions of Solidity are used:
- Version used: ['>=0.5.0', '>=0.6.2', '^0.8.0']
- >=0.5.0 (contracts/v2-core/contracts/interfaces/IUniswapV2Factory.sol#1)
- >=0.6.2 (contracts/v2-periphery/contracts/interfaces/IUniswapV2Router01.sol#1)
- >=0.6.2 (contracts/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol#1)
- ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4)
- ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
- ^0.8.0 (contracts/Gochan.sol#2)
- ^0.8.0 (contracts/openzeppelin-contracts/contracts/access/Ownable.sol#4)
- ^0.8.0 (contracts/openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#4)
- ^0.8.0 (contracts/openzeppelin-contracts/contracts/token/ERC20/IERC20.sol#4)
- ^0.8.0 (contracts/openzeppelin-contracts/contracts/token/ERC20/extensions/IERC20Metadata.sol#4)
- ^0.8.0 (contracts/openzeppelin-contracts/contracts/utils/Context.sol#4)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

ERC20._mint(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#251-264) has costly operations inside a loop:
- _totalSupply += amount (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#256)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

Context._msgData() (contracts/openzeppelin-contracts/contracts/utils/Context.sol#21-23) is never used and should be removed
ERC20._burn(address,uint256) (contracts/openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#285-301) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

allows old versions
Pragma version^0.8.0 (contracts/openzeppelin-contracts/contracts/utils/Context.sol#4) allows old versions
Pragma version>=0.5.0 (contracts/v2-core/contracts/interfaces/IUniswapV2Factory.sol#1) allows old versions
Pragma version>=0.6.2 (contracts/v2-periphery/contracts/interfaces/IUniswapV2Router01.sol#1) allows old versions
Pragma version>=0.6.2 (contracts/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol#1) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Snapshot 0.3

```
Gochan._spendAllowance(address,address,uint256).owner (contracts/Gochan.sol#175) shadows:
	- Ownable.owner() (node_modules/@openzeppelin/contracts/access/Ownable.sol#43-45) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Gochan.constructor(address,address,address[],uint256[],uint256)._router (contracts/Gochan.sol#36) lacks a zero-check on :
		- routerAddress = _router (contracts/Gochan.sol#46)
Gochan.constructor(address,address,address[],uint256[],uint256)._taxReceiver (contracts/Gochan.sol#37) lacks a zero-check on :
		- recipient = _taxReceiver (contracts/Gochan.sol#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in Gochan.addInitialLiquidity(uint256,uint256) (contracts/Gochan.sol#73-100):
	External calls:
	- pair = IUniswapV2Factory(router.factory()).createPair(address(this),router.WETH()) (contracts/Gochan.sol#83-86)
	State variables written after the call(s):
	- dex[pair] = true (contracts/Gochan.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in Gochan._transfer(address,address,uint256) (contracts/Gochan.sol#102-172):
	External calls:
	- _swapTokensToEth(toSwap,recipient) (contracts/Gochan.sol#168)
		- router.swapExactTokensForETH(tokenAmount,0,path,recip,block.timestamp + 900) (contracts/Gochan.sol#193-199)
	Event emitted after the call(s):
	- Transfer(from,to,amount) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#237)
		- super._transfer(from,to,toRecip) (contracts/Gochan.sol#171)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Gochan.addInitialLiquidity(uint256,uint256) (contracts/Gochan.sol#73-100) uses timestamp for comparisons
	Dangerous comparisons:
	- require(bool,string)(timeStart == 0,Liquidity already added) (contracts/Gochan.sol#78)
Gochan._transfer(address,address,uint256) (contracts/Gochan.sol#102-172) uses timestamp for comparisons
	Dangerous comparisons:
	- blockStart == 0 || timeStart == 0 (contracts/Gochan.sol#109)
	- isDuringFx = block.timestamp <= timeStart + fxEnd (contracts/Gochan.sol#119)
	- ! isBeforeFx && isDuringFx (contracts/Gochan.sol#136)
	- require(bool,string)(block.timestamp - lastTxTimestamp[from] >= txCooldown,Transfer cooldown not expired.) (contracts/Gochan.sol#143-146)
	- ! dex[from] && taxCollected > 0 && block.timestamp - lastTaxBlock > taxThreshold (contracts/Gochan.sol#161-163)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

Auto

Reentrancy in Gochan._transfer
uses a dangerous strict equality:
- blockStart == 0 || timeStart == 0 (contracts/Gochan.sol#109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in Gochan.addInitialLiquidity
router.addLiquidityETH{value: ethAmount}(address(this),liquidityAmount,0,0,msg.sender,block.timestamp + 900) (contracts/Gochan.sol#89-96)
State variables written after the call(s):
- blockStart = block.number (contracts/Gochan.sol#98)
Gochan.blockStart (contracts/Gochan.sol#14) can be used in cross function reentrancies:
- Gochan._transfer(address,address,uint256) (contracts/Gochan.sol#102-172)
- Gochan.addInitialLiquidity(uint256,uint256) (contracts/Gochan.sol#73-100)
- Gochan.blockStart (contracts/Gochan.sol#14)
- timeStart = block.timestamp (contracts/Gochan.sol#99)
Gochan.timeStart (contracts/Gochan.sol#15) can be used in cross function reentrancies:
- Gochan._transfer(address,address,uint256) (contracts/Gochan.sol#102-172)
- Gochan.addInitialLiquidity(uint256,uint256) (contracts/Gochan.sol#73-100)
- Gochan.timeStart (contracts/Gochan.sol#15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

Context is re-used
- Context (node_modules/@openzeppelin/contracts/utils/Context.sol#16-24)
- Context (contracts/openzeppelin-contracts/contracts/utils/Context.sol#16-24)

0.2 SOLIDITY UNIT TESTING

Progress: Starting
PASS ✓ **Tested**

✓ Check winning proposal
✓ Check winnin proposal with return value
✓ Before all
✓ Check success
✓ Check success2
✓ Check sender and value

Result for tests Passed:

0Time Taken: 0.54s

Manual and Automated Vulnerability Test

CRITICAL ISSUES

During the audit, AudiTBlock experts found **0 medium Critical issues** in the code of the smart contract.

HIGH ISSUES

During the audit, AudiTBlock experts found **0 High issues** in the code of the smart contract.

MEDIUM ISSUES

During the audit, AudiTBlock experts found **2 Medium issues** in the code of the smart contract.

LOW ISSUES

During the audit, AudiTBlock experts found **0 Low issues** in the code of the smart contract.

INFORMATIONAL ISSUES

During the audit, AuditBlock experts found **2 Informational issues** in the code of the smart contract.

SWC Attacks

ID	Title		Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	✔
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✔
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✔
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	✔
SWC-127	Arbitrary Jump with Function TypeVariable	CWE-695: Use of Low-Level Functionality	✔
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✔
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✔
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✔

ID	Title		Test Result
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✔
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✔
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✔
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✔
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✔
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✔
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✔
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✔
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✔
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✔

Owner privileges

- Status: tested 1 and verified✓
- Status: tested 2 and verified✓
- Status: tested 3 and verified✓
- Status: tested 4 and verified✓

Executive Summary

Two (2) independent AuditBlock experts performed an unbiased and isolated audit of the smart contract. The final debriefs

The overall code quality is good and not overloaded with unnecessary functions, these is greatly

benefiting the security of the contract. It correctly implemented widely used and reviewed contracts he main goal of the audit was to verify the claims regarding the security of the smart contract and the claims inside the scope of work.

During the audit, no Critical issues were found after the manual and automated security testing.

Deployed On Etherscan.io

VERIFIED ✔

https://etherscan.io/token/0x92f702a11cd98cf05cf5947c39ff5b4ce06c2099#code