Cyberscope

Audit Report

Invest Club Global

January 2024

Network	ETH
Address	0x9F9643209dCCe8D7399D7BF932354768069Ebc64
Audited by	© cyberscope

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

🔴 Critical 🟠 Medium ● Minor / Informative

Severity	Code	Description	Status
●	DDP	Decimal Division Precision	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved
●	L19	Stable Compiler Version	Unresolved
●	L22	Potential Locked Ether	Unresolved

Table of Contents

Review

Contract Name	InvestClubGlobal
Compiler Version	v0.8.20+commit.a1b79de6
Optimization	200 runs
Explorer	https://etherscan.io/address/0x9f9643209dcce8d7399d7bf932354768069ebc64
Address	0x9f9643209dcce8d7399d7bf932354768069ebc64
Network	ETH
Symbol	ICG
Decimals	18
Total Supply	900,000,000,000

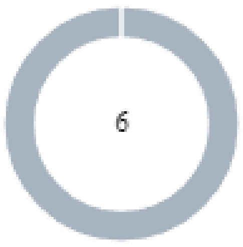
Audit Updates

Initial Audit	02 Jan 2024

Source Files

Filename	SHA256
InvestClubGlobal.sol	a54c1dece960cf9e44eb1602ced271d222b3e53bede27df603907c6c91e59d6e

Findings Breakdown



●	Critical	0
●	Medium	0
●	Minor / Informative	6

Severity	Unresolved	Acknowledged	Resolved	Other
● Critical	0	0	0	0
● Medium	0	0	0	0
● Minor / Informative	6	0	0	0

DDP - Decimal Division Precision

Criticality	Minor / Informative
Location	InvestClubGlobal.sol#L1185,1187
Status	Unresolved

Description

Division of decimal (fixed point) numbers can result in rounding errors due to the way that division is implemented in Solidity. Thus, it may produce issues with precise calculations with decimal numbers.

Solidity represents decimal numbers as integers, with the decimal point implied by the number of decimal places specified in the type (e.g. decimal with 18 decimal places). When a division is performed with decimal numbers, the result is also represented as an integer, with the decimal point implied by the number of decimal places in the type. This can lead to rounding errors, as the result may not be able to be accurately represented as an integer with the specified number of decimal places.

Hence, the splitted shares will not have the exact precision and some funds may not be calculated as expected.

```
if (takeFee) {
    uint256 feeAmt;
    if (automatedMarketMakerPairs[to]) {
        feeAmt = (amount * totalSellTax) / 100;
    } else if (automatedMarketMakerPairs[from]) {
        feeAmt = (amount * totalBuyTax) / 100;
    }
    amount = amount - feeAmt;
    super._transfer(from, marketingWallet, feeAmt);
}
```

Recommendation

The team is advised to take into consideration the rounding results that are produced from the solidity calculations. The contract could calculate the subtraction of the divided funds in the last calculation in order to avoid the division rounding issue.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	InvestClubGlobal.sol#L1009,1022
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
maxWalletAmount = (totalSupply() * maxWalletPercent) / 100;

maxBuyAmount = (totalSupply() * maxBuyPercent) / 100;
maxSellAmount = (totalSupply() * maxSellPercent) / 100;
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	InvestClubGlobal.sol#L195,290,303,315
Status	Unresolved

Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function _contextSuffixLength() internal view virtual returns
(uint256) {
    return 0;
}

function _requirePaused() internal view virtual {
    if (!paused()) {
        revert ExpectedPause();
    }
}

...
```

Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	InvestClubGlobal.sol#L1170,1183
Status	Unresolved

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
bool takeFee;
uint256 feeAmt;
```

Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	InvestClubGlobal.sol#L9,174,205,326,428,510,538,856,891,939
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.20;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

L22 - Potential Locked Ether

Criticality	Minor / Informative
Location	InvestClubGlobal.sol#L989
Status	Unresolved

Description

The contract contains Ether that has been placed into a Solidity contract and is unable to be transferred. Thus, it is impossible to access the locked Ether. This may produce a financial loss for the users that have called the payable method.

```
receive() external payable {}
```

Recommendation

The team is advised to either remove the payable method or add a withdraw functionality. it is important to carefully consider the risks and potential issues associated with locked Ether.

Functions Analysis

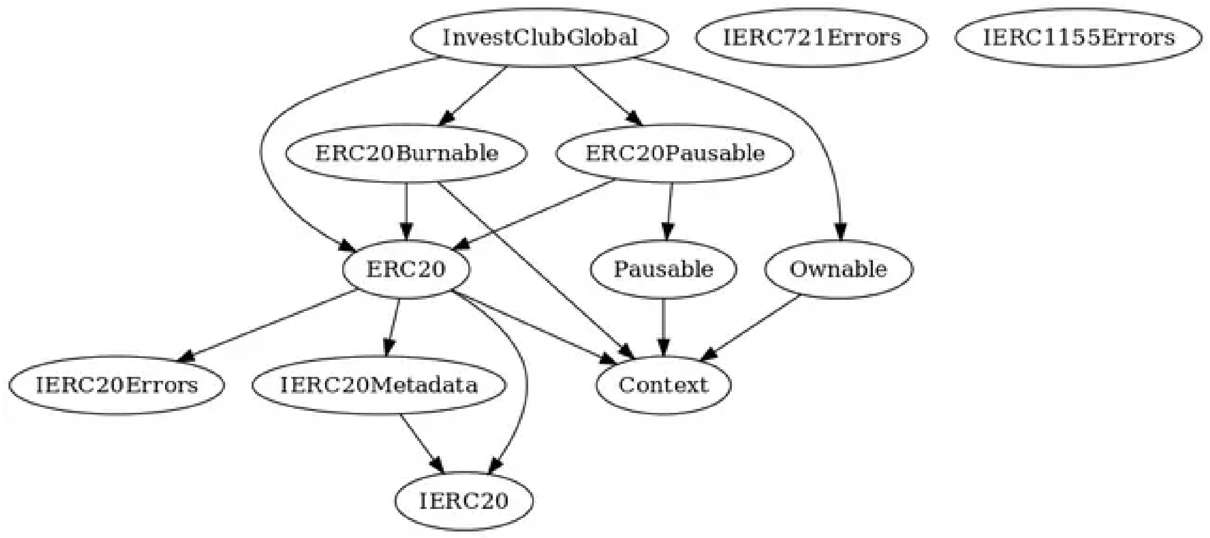
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IERC20Errors	Interface			
IERC721Errors	Interface			
IERC1155Errors	Interface			
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
	_contextSuffixLength	Internal		
Pausable	Implementation	Context		
		Public	✓	-
	paused	Public		-
	_requireNotPaused	Internal		
	_requirePaused	Internal		
	_pause	Internal	✓	whenNotPaused
	_unpause	Internal	✓	whenPaused

Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
ERC20	Implementation	Context, IERC20, IERC20Metadata,		

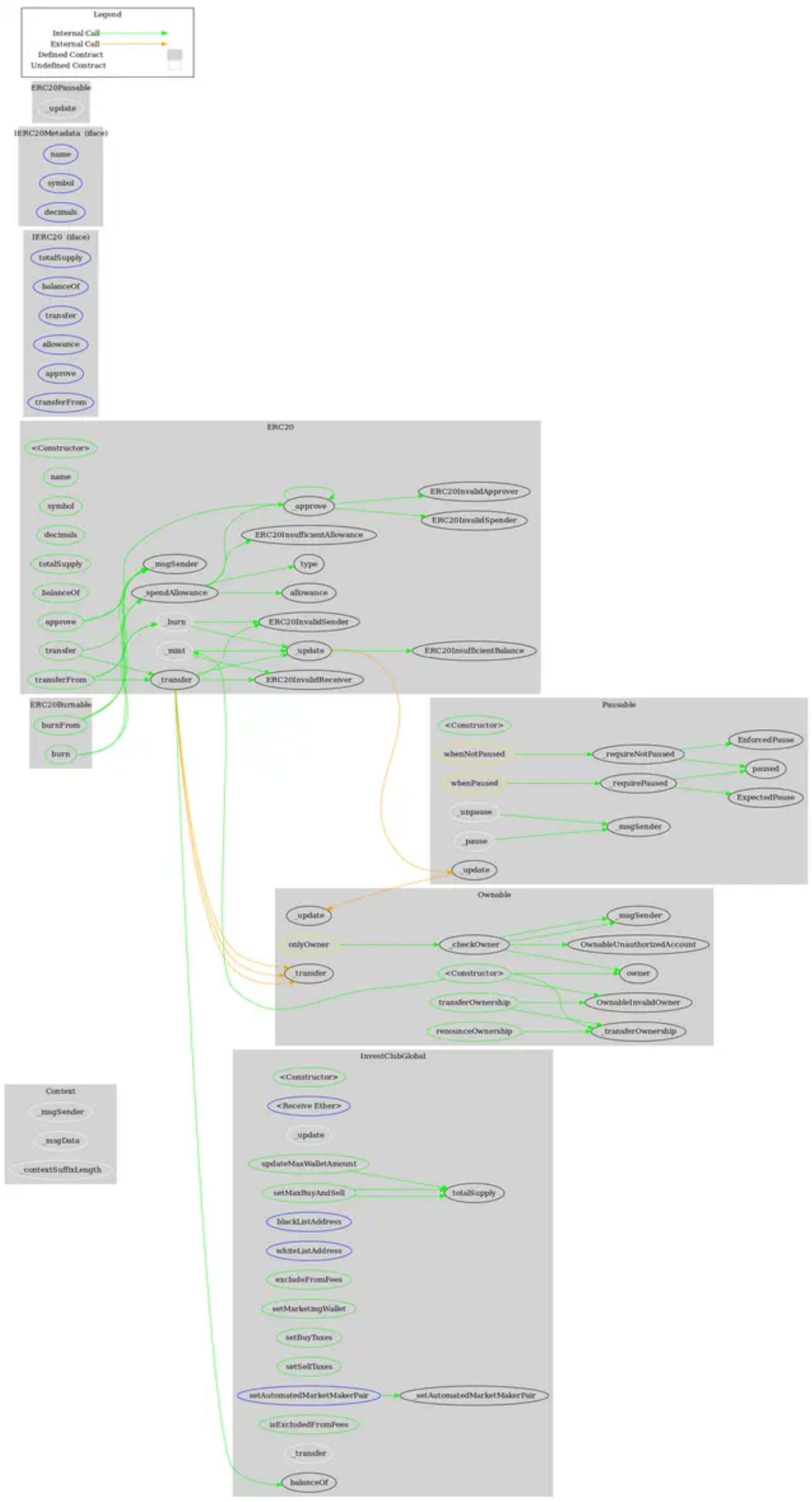
		IERC20Errors		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	_transfer	Internal	✓	
	_update	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
ERC20Pausable	Implementation	ERC20, Pausable		
	_update	Internal	✓	whenNotPaused
ERC20Burnable	Implementation	Context, ERC20		

	burn	Public	✓	-
	burnFrom	Public	✓	-
InvestClubGlobal	Implementation	ERC20, ERC20Burnable, ERC20Pausable, Ownable		
		Public	✓	Ownable ERC20
		External	Payable	-
	_update	Internal	✓	
	updateMaxWalletAmount	Public	✓	onlyOwner
	setMaxBuyAndSell	Public	✓	whenNotPaused onlyOwner
	blackListAddress	External	✓	onlyOwner
	whiteListAddress	External	✓	whenNotPaused onlyOwner
	excludeFromFees	Public	✓	whenNotPaused onlyOwner
	setMarketingWallet	Public	✓	whenNotPaused onlyOwner
	setBuyTaxes	Public	✓	whenNotPaused onlyOwner
	setSellTaxes	Public	✓	whenNotPaused onlyOwner
	setAutomatedMarketMakerPair	External	✓	whenNotPaused onlyOwner
	_setAutomatedMarketMakerPair	Private	✓	
	isExcludedFromFees	Public		-
	_transfer	Internal	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
ERC20Pausable
_update
IERC20Metadata (iface)
name
symbol
decimals
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
approve
transfer
transferFrom
_msgSender
_spendAllowance
_burn
_mint
_transfer
_approve
ERC20InsufficientAllowance
_type
allowance
ERC20InvalidSender
_update
ERC20InvalidReceiver
ERC20InvalidApprover
ERC20InvalidSpender
ERC20InsufficientBalance
ERC20Burnable
burnFrom
burn
Pausable
<Constructor>
whenNotPaused
whenPaused
_unpause
_pause
_update
_requireNotPaused
_requirePaused
_msgSender
EnforcedPause
paused
ExpectedPause
Ownable
_update
onlyOwner
_transfer
_checkOwner
<Constructor>
transferOwnership
renounceOwnership
_msgSender
OwnableUnauthorizedAccount
owner
OwnableInvalidOwner
_transferOwnership
Context
_msgSender
_msgData
_contextSuffixLength
InvestClubGlobal
<Constructor>
<Receive Ether>
_update
updateMaxWalletAmount
setMaxBuyAndSell
blackListAddress
whiteListAddress
excludeFromFees
setMarketingWallet
setBuyTaxes
setSellTaxes
setAutomatedMarketMakerPair
isExcludedFromFees
_transfer
balanceOf
totalSupply
_setAutomatedMarketMakerPair

Summary

Invest Club Global contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. Invest Club Global is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. 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 Cyberscope's prior written consent. 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 regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document 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 report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment 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.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io