



Cyberscope

# Audit Report

## **BlockATM**

June 2023

Audited by © cyberscope

# Table of Contents

<b>Table of Contents</b>	<b>1</b>
<b>Review</b>	<b>3</b>
Audit Updates	3
Source Files	3
<b>Introduction</b>	<b>4</b>
Roles	5
BlockATM	5
Owner	5
User	5
BlockATMCustomer	6
Owner	6
User	6
BlockOrder	7
Owner	7
User	7
BlockRecharge	8
Owner	8
User	8
Test Deployments	9
<b>Findings Breakdown</b>	<b>10</b>
Findings	11
OCTD - Transfers Contract's Tokens	12
Description	12
Recommendation	12
Team Update	13
RSML - Redundant SafeMath Library	14
Description	14
Recommendation	14
L11 - Unnecessary Boolean equality	15
Description	15
Recommendation	15
<b>Function Analysis</b>	<b>16</b>
<b>Flow Graph</b>	<b>19</b>
<b>Summary</b>	<b>20</b>
<b>Disclaimer</b>	<b>21</b>
<b>About Cyberscope</b>	<b>22</b>

# Review

## Audit Updates

Initial Audit	24 May 2023 <a href="https://github.com/cyberscope-io/audits/blob/main/batm/v1/audit.pdf">https://github.com/cyberscope-io/audits/blob/main/batm/v1/audit.pdf</a>
Corrected Phase 2	08 Jun 2023

## Source Files

Filename	SHA256
BlockATM.sol	456418f9ad38efd140eedb470846f67a1ac4594773bd25287ba5dbdec2e54613
BlockATMCustomer.sol	a817ffb33f562e14fd737819492f21dcb3e3685dcf729d8b8648008c0d116044
BlockCommon.sol	65be290bd07296955de7a5ef05dca23b1c1446bfd04db8d6ec89818e62a07821
BlockOrder.sol	9ca07ace1a1fc5e78ace9c79cb12c76ecdd5589b33f375d2a0e3a4b68b771a1a
BlockPlatform.sol	6fe105db2769330ac8c252e8282baa18dfba22f1b1908b6eaab434d77dc73d5b
BlockPlatformStandard.sol	1f374e28659da11ea2090fc3f2e3b81567e09c485a58f702962ca05bfe3b88c6
BlockRecharge.sol	511045d43ef79eab52ac7b833d475d6cca59a5d5133fbf8d3fc576f0366f1901

# Introduction

The BlockATM ecosystem consists of four distinct contracts: BlockATM, BlockATMCustomer, BlockOrder, and BlockRecharge. The BlockATM contract serves as an automated teller machine for cryptocurrencies, enabling users to transfer ERC20 tokens and Ether (ETH) to a designated withdrawal address. The BlockATMCustomer contract is designed for customers of the BlockATM system, allowing them to transfer ERC20 tokens and subsequently withdraw them to specified addresses. BlockOrder facilitates the transfer of ERC20 tokens associated with orders or specific transactions to a designated withdrawal address. Lastly, BlockRecharge offers the same functionality as the BlockOrder contract. These contracts provide various features and functionality related to token transfers and withdrawals, with the contract owners having control over withdrawal addresses, supported tokens, and other related settings.

## Roles

### BlockATM

#### Owner

The owner has authority over the following functions:

- `function modifyWithdrawAddress(address payable newAddress)`
- `function addCoinList(address _address)`
- `function closeCoinList(address newAddress)`

#### User

The user can interact with the following functions:

- `function transferToken(address token,uint256 amount,string memory orderId)`
- `function transferETH(string memory orderId)`
- `function getWithdrawAddress()`
- `function getCoinList(address _address)`

## BlockATMCustomer

### Owner

The owner has authority over the following functions:

- `function withdrawToken(uint256 amount,address withdrawAddress)`
- `function setActiveFlag()`

### User

The user can interact with the following functions:

- `function transferToken(uint256 amount,string memory orderId)`
- `function getTokenAddress()`
- `function getActiveFlag()`
- `function getWithdrawAddressList()`
- `function checkWithdrawAddress(address withdrawAddress)`
- `function getWithdrawAddressFlag(address withdrawAddress)`
- `function getOwnerAddressFlag(address ownerAddress))`
- `function getOwnerAddressList()`

## BlockOrder

### Owner

The owner has authority over the following functions:

- `function modifyWithdrawAddress(address payable _address)`
- `function addCoinList(address _address)`
- `function closeCoinList(address _address)`

### User

The user can interact with the following functions:

- `function transferToken(address token,uint256 amount,string memory orderId)`
- `function getWithdrawAddress()`
- `function getCoinList(address _address)`

## BlockRecharge

### Owner

The owner has authority over the following functions:

- `function modifyWithdrawAddress(address payable _address)`
- `function addCoinList(address _address)`
- `function closeCoinList(address _address)`

### User

The user can interact with the following functions:

- `function transferToken(address token,uint256 amount,string memory orderId)`
- `function getWithdrawAddress()`
- `function getCoinList(address _address)`



## Test Deployments

Contract	Explorer
BlockATM	<a href="https://testnet.bscscan.com/address/0xbf49dE2941335157874a32c4a84054fC6312E838">https://testnet.bscscan.com/address/0xbf49dE2941335157874a32c4a84054fC6312E838</a>
BlockATMCustomer	<a href="https://testnet.bscscan.com/address/0x30D7D291AFF1fF3Bc843fb5f6eb142EB97631725">https://testnet.bscscan.com/address/0x30D7D291AFF1fF3Bc843fb5f6eb142EB97631725</a>
BlockOrder	<a href="https://testnet.bscscan.com/address/0x13d9B1A369137a45D86Ab4f161716E0d64cA9207">https://testnet.bscscan.com/address/0x13d9B1A369137a45D86Ab4f161716E0d64cA9207</a>
BlockRecharge	<a href="https://testnet.bscscan.com/address/0xb1d6eA6d21C7Fe7cA02824cc5732e914128504f1">https://testnet.bscscan.com/address/0xb1d6eA6d21C7Fe7cA02824cc5732e914128504f1</a>

## Findings Breakdown



● Critical	0
● Medium	0
● Minor / Informative	3

Severity	Unresolved	Acknowledged	Resolved	Other
● Critical	0	0	0	0
● Medium	0	0	0	0
● Minor / Informative	2	1	0	0

## Findings

Severity	Code	Description	Status
●	OCTD	Transfers Contract's Tokens	Acknowledged
●	RSML	Redundant SafeMath Library	Unresolved
●	L11	Unnecessary Boolean equality	Unresolved

## OCTD - Transfers Contract's Tokens

Criticality	Minor / Informative
Location	BlockATMCustomer.sol#L54
Status	Acknowledged

### Description

The contract owner has the authority to claim all the balance of the contract. The balance of the contract is the accumulated amount of all the users' deposits. The owner may take advantage of it by adding only the owner's address to the `ownerMap` variable and then calling the `withdrawToken` function.

```
modifier onlyOwner() {
    require(ownerMap[msg.sender] == 1, "Not the owner");
    _;
}

function withdrawToken(uint256 amount, address withdrawAddress) public
onlyOwner returns (bool) {
    // check withdrawAddress
    require(withdrawMap[withdrawAddress], "withdraw address not allowed");
    super.withdrawCommon(tokenAddress, withdrawAddress, amount);
    emit WithdrawToken(msg.sender, withdrawAddress, tokenAddress, amount);
    return true;
}
```

### Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

- Renouncing the ownership will eliminate the threats but it is non-reversible.

## Team Update

According to business requirements, the ownerMap needs to have multiple owners managing. And the owner is added when the contract is deployed, it cannot be modified again. No adjustment will be made.

## RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	BlockCommon.sol
Status	Unresolved

### Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert to underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases gas consumption unnecessarily.

```
library SafeMath {...}
```

### Recommendation

The team is advised to remove the SafeMath library. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the `unchecked { ... }` statement.

Read more about the breaking change on

<https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes>.

## L11 - Unnecessary Boolean equality

Criticality	Minor / Informative
Location	BlockPlatform.sol#L32
Status	Unresolved

### Description

Boolean equality is unnecessary when comparing two boolean values. This is because a boolean value is either true or false, and there is no need to compare two values that are already known to be either true or false.

it's important to be aware of the types of variables and expressions that are being used in the contract's code, as this can affect the contract's behavior and performance. The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
require (coinList[newAddress] == false, "The token address is approved")
```

### Recommendation

Using the boolean value itself is clearer and more concise, and it is generally considered good practice to avoid unnecessary boolean equalities in Solidity code.

## Function Analysis

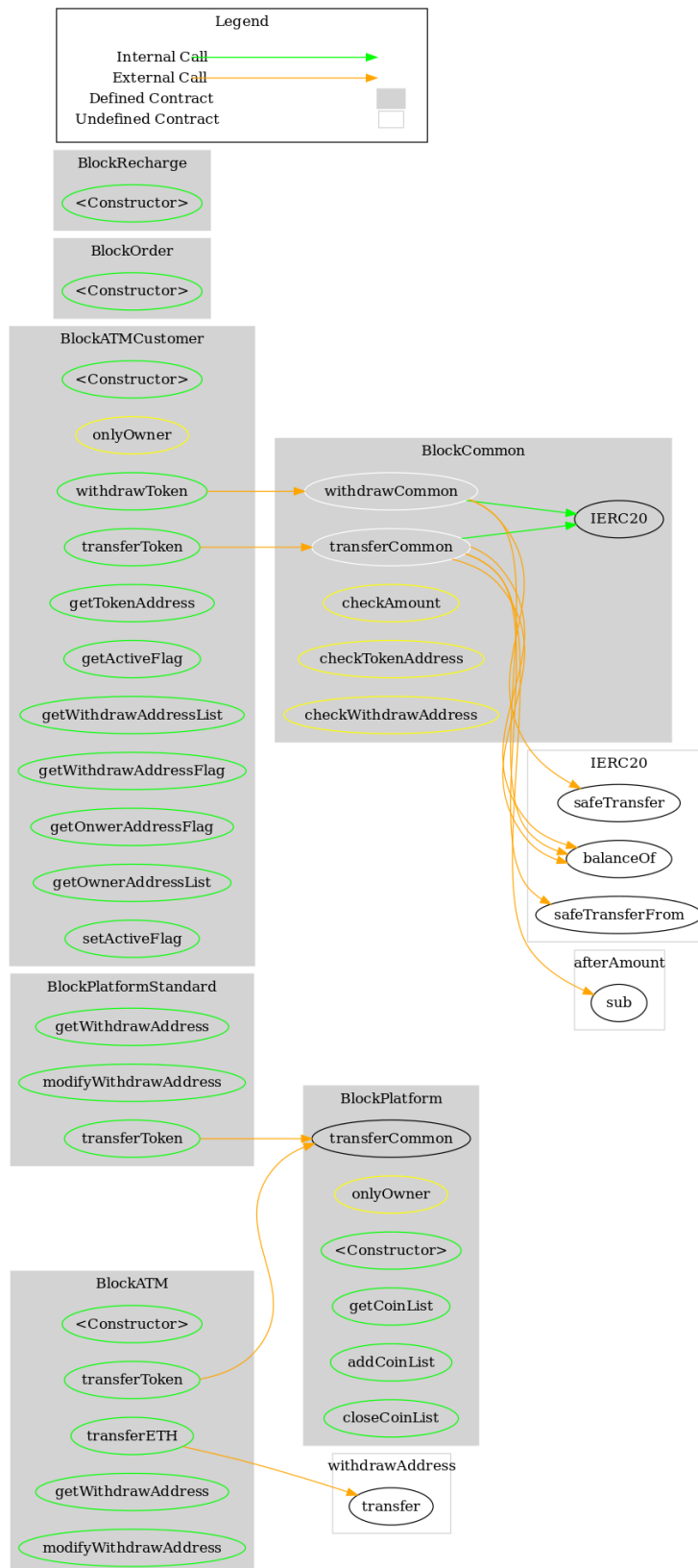
Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
<b>BlockATM</b>	Implementation	BlockPlatform		
		Public	✓	checkWithdrawAddress
	transferToken	Public	✓	-
	transferETH	Public	Payable	checkAmount
	getWithdrawAddress	Public		-
	modifyWithdrawAddress	Public	✓	onlyOwner checkWithdrawAddress
<b>BlockATMCustomer</b>	Implementation	BlockCommon		
		Public	✓	checkTokenAddress
	transferToken	Public	✓	-
	withdrawToken	Public	✓	onlyOwner
	getTokenAddress	Public		-
	getActiveFlag	Public		-
	getWithdrawAddressList	Public		-
	getWithdrawAddressFlag	Public		-
	getOwnerAddressFlag	Public		-
	getOwnerAddressList	Public		-



	setActiveFlag	Public	✓	onlyOwner
<b>BlockCommon</b>	Implementation			
	transferCommon	Internal	✓	checkTokenAddress checkAmount
	withdrawCommon	Internal	✓	checkAmount checkTokenAddress checkWithdrawAddress
<b>BlockOrder</b>	Implementation	BlockPlatformStandard		
		Public	✓	checkWithdrawAddress
<b>BlockPlatform</b>	Implementation	BlockCommon		
		Public	✓	-
	getCoinList	Public		-
	addCoinList	Public	✓	onlyOwner checkTokenAddress
	closeCoinList	Public	✓	onlyOwner checkTokenAddress
<b>BlockPlatformStandard</b>	Implementation	BlockPlatform		
	getWithdrawAddress	Public		-
	modifyWithdrawAddress	Public	✓	onlyOwner checkWithdrawAddress
	transferToken	Public	✓	-

<b>BlockRecharge</b>	Implementation	BlockPlatformStandard		
		Public	✓	checkWithdrawAddress

# Flow Graph



## Summary

BlockATM contract implements a financial mechanism. This audit investigates security issues, business logic concerns, and potential improvements.

## 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>