Cyberscope

Audit Report

BlockATM

June 2023

Audited by © cyberscope

Table of Contents

Review

Audit Updates

Initial Audit	24 May 2023 https://github.com/cyberscope-io/audits/blob/main/batm/v1/audit.pdf
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 getOnwerAddressFlag(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

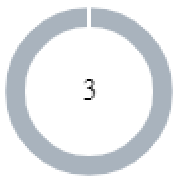
The user can interact with the following functions:

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

Test Deployments

Contract	Explorer
BlockATM	https://testnet.bscscan.com/address/0xbf49dE2941335157874a32c4a84054fC6312E838
BlockATMCustomer	https://testnet.bscscan.com/address/0x30D7D291AFF1fF3Bc843fb5f6eb142EB97631725
BlockOrder	https://testnet.bscscan.com/address/0x13d9B1A369137a45D86Ab4f161716E0d64cA9207
BlockRecharge	https://testnet.bscscan.com/address/0xb1d6eA6d21C7Fe7cA02824cc5732e914128504f1

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

he contract owner has the authority to claim all the balance of the contract. The balance of 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.

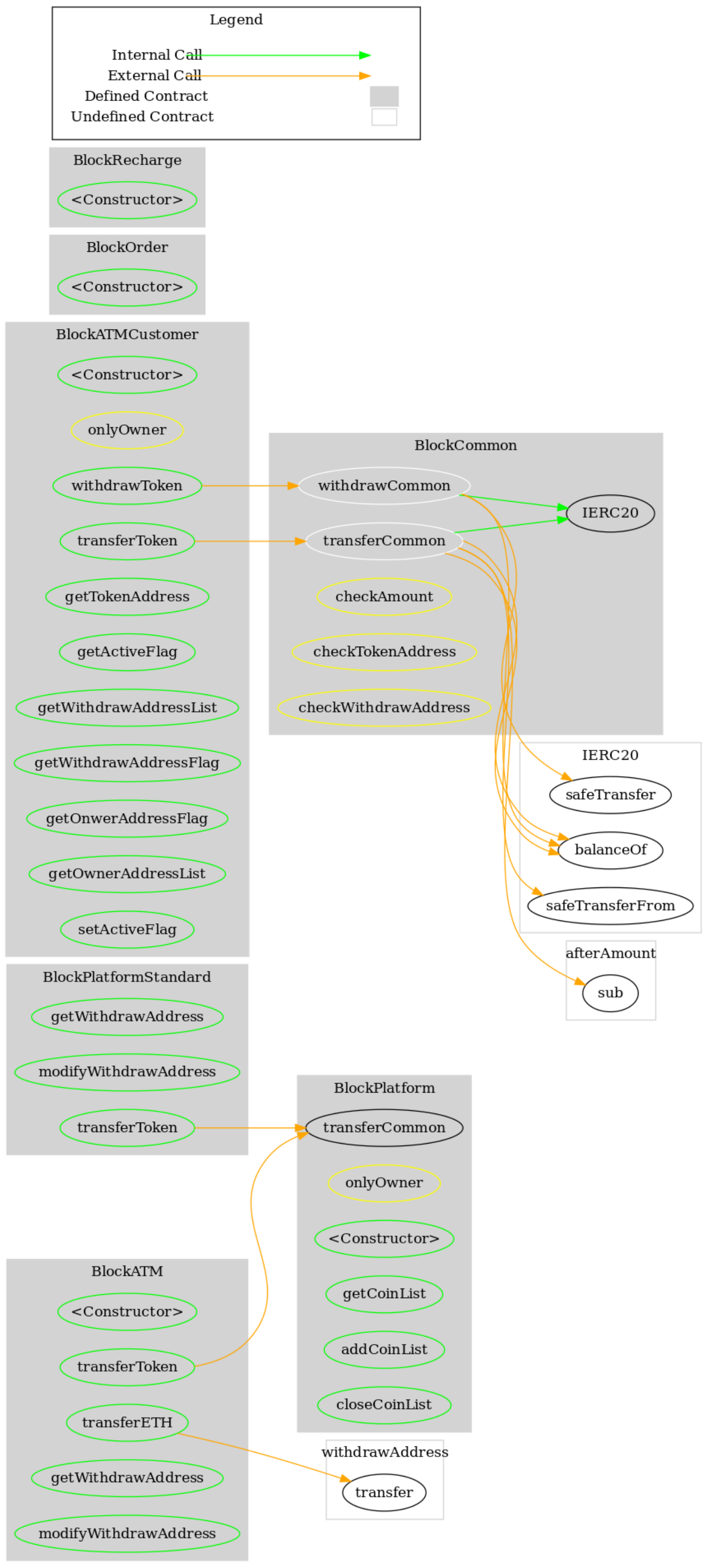
Function Analysis

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

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

BlockRecharge	Implementation	BlockPlatformStandard		
		Public	✓	checkWithdrawAddress

Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
BlockRecharge
<Constructor>
BlockOrder
<Constructor>
BlockATMCustomer
<Constructor>
onlyOwner
withdrawToken
transferToken
getTokenAddress
getActiveFlag
getWithdrawAddressList
getWithdrawAddressFlag
getOnwerAddressFlag
getOwnerAddressList
setActiveFlag
BlockCommon
withdrawCommon
transferCommon
checkAmount
checkTokenAddress
checkWithdrawAddress
IERC20
IERC20
safeTransfer
balanceOf
safeTransferFrom
afterAmount
sub
BlockPlatformStandard
getWithdrawAddress
modifyWithdrawAddress
transferToken
BlockPlatform
transferCommon
onlyOwner
<Constructor>
getCoinList
addCoinList
closeCoinList
BlockATM
<Constructor>
transferToken
transferETH
getWithdrawAddress
modifyWithdrawAddress
withdrawAddress
transfer

Summary

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

Disclaimer

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