Cyberscope

Audit Report

BlaroThings

April 2025

marketplace.sol	dc728ba5d2b6e0c11e954626fa196d7224ea1b4877c969cd44e02bb3b4c6cea9
investmentPool.sol	f4222cb8401803ea5df3ead6b8be59d346776b8fe058c05cda65303407926123
FactoryPool.sol	6a687561287dc1871558c763d3569529bd6e3149471badf09a1b31c9608ed100

Audited by © cyberscope

Table of Contents

Risk Classification

The criticality of findings in Cyberscope's smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation**: This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation**: This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical**: Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium**: Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor**: Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative**: Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

Severity	Likelihood / Impact of Exploitation
Critical	Highly Likely / High Impact
Medium	Less Likely / High Impact or Highly Likely/ Lower Impact
Minor / Informative	Unlikely / Low to no Impact

Review

Audit Updates

Initial Audit	13 Jan 2025
Corrected Phase 2	23 Jan 2025
Corrected Phase 3	11 Mar 2025
Corrected Phase 4	08 Apr 2025
Corrected Phase 5	22 Apr 2025

Source Files

Filename	SHA256
marketplace.sol	dc728ba5d2b6e0c11e954626fa196d7224ea1b4877c969cd44e02bb3b4c6cea9
investmentPool.sol	f4222cb8401803ea5df3ead6b8be59d346776b8fe058c05cda65303407926123
FactoryPool.sol	6a687561287dc1871558c763d3569529bd6e3149471badf09a1b31c9608ed100

Overview

The BlaroThings Ecosystem consists of several interconnected smart contracts. This audit focuses on the InvestmentPool, PoolFactory, and Marketplace that together provide a comprehensive platform for secure investments, pool management, and secondary trading of investment shares. These contracts work seamlessly to enable users to participate in investment opportunities, manage pools, and trade shares in a decentralized and efficient manner.

InvestmentPool

The InvestmentPool contract is at the core of the ecosystem, providing a decentralized mechanism for managing investments in approved stablecoins. Users can deposit stablecoins into a pool with a defined maximum capacity and lifespan, receiving shares proportional to their contributions. The pool closes to new investments once it reaches its capacity or maturity date, after which users can withdraw their funds.

Key features include:

- **Early Withdrawals:** Investors can withdraw their funds before maturity, incurring a predefined penalty.
- **Profit Distribution:** Pool owners can distribute profits to investors based on their shareholdings.
- **Administrative Control:** Pool owners can manage approved stablecoins, close pools, and distribute profits securely.

The contract ensures robust security with reentrancy protection and strict access controls, providing a reliable environment for investments.

PoolFactory

The PoolFactory contract acts as a factory for creating and managing multiple instances of InvestmentPool. It streamlines the deployment of investment pools by allowing the contract owner to configure key parameters such as capacity, lifespan, and early withdrawal penalties.

Features include:

- **Pool Creation:** Deploys new pools and tracks them in a structured manner.
- **Pool Management:** Enables verification and interaction with registered pools to ensure integrity.
- **Administrative Control:** Allows the owner to close pools, ensuring lifecycle management.

By simplifying the creation and management of pools, the PoolFactory provides a scalable and secure way to manage multiple investment opportunities.

Marketplace

The Marketplace contract facilitates a secondary market for trading shares in InvestmentPool contracts. Using the native BLR token, users can list, buy, and delist shares, enabling liquidity and flexibility within the ecosystem. The marketplace charges a trading fee, which is accumulated and managed by the contract owner.

Key functionalities:

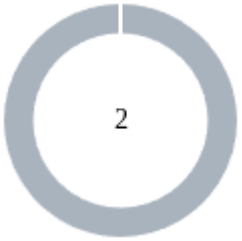
1. **Listing Shares:** Sellers can list shares from specific pools for sale with a specified price per share.
2. **Purchasing Shares:** Buyers can acquire listed shares, with the contract handling payment and fee distribution.
3. **Delisting Shares:** Sellers can partially or fully remove their unsold shares from listings.
4. **Fee Management:** The contract collects a trading fee on each purchase, which is withdrawn by the owner to support operations.

The Marketplace ensures smooth integration with InvestmentPool contracts, though certain aspects (e.g., reliance on specific functions) require careful alignment with the pools' intended behavior.

Summary

The BlaroThings Ecosystem provides a robust framework for decentralized investments and trading. InvestmentPool enables secure and managed investment operations, PoolFactory simplifies the deployment and management of multiple pools, and Marketplace enhances liquidity by offering a secondary trading platform for shares. Together, these contracts deliver a comprehensive and integrated solution for decentralized financial ecosystems.

Findings Breakdown



Critical	0
Medium	0
Minor / Informative	2

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

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	MSP	Missing Stablecoin Payout	Unresolved
	L05	Unused State Variable	Unresolved

MSP - Missing Stablecoin Payout

Criticality	Minor / Informative
Location	marketplace.sol#L120
Status	Unresolved

Description

The `purchaseShares` function correctly debits BLR tokens from the buyer and pays out the seller's share, but never transfers the purchased stablecoins to the buyer. As a result, users swap away their BLR and receive nothing in return, violating the intended exchange logic and risking user fund loss.

```
function purchaseShares(
    address pool,
    uint256 listingIndex,
    uint256 amount
) external nonReentrant {
    ...
    // First transfer the total amount from buyer to contract
    require(
        IBlrToken(blrToken).transferFrom(msg.sender, address(this), totalPrice),
        "Transfer from buyer failed"
    );

    // Then transfer the seller's portion to the seller
    require(
        IBlrToken(blrToken).transfer(listing.seller, sellerAmount),
        "Transfer to seller failed"
    );
    ...
}
```

Recommendation

The team is advised to modify the function and ensure the purchased funds are correctly transferred to the buyer. The contract should also validate the transfer's success to guarantee that users receive their purchased shares.

L05 - Unused State Variable

Criticality	Minor / Informative
Location	investmentPool.sol#L23
Status	Unresolved

Description

An unused state variable is a state variable that is declared in the contract, but is never used in any of the contract's functions. This can happen if the state variable was originally intended to be used, but was later removed or never used.

Unused state variables can create clutter in the contract and make it more difficult to understand and maintain. They can also increase the size of the contract and the cost of deploying and interacting with it.

```
uint8 private constant STANDARD_DECIMALS = 18
```

Recommendation

To avoid creating unused state variables, it's important to carefully consider the state variables that are needed for the contract's functionality, and to remove any that are no longer needed. This can help improve the clarity and efficiency of the contract.

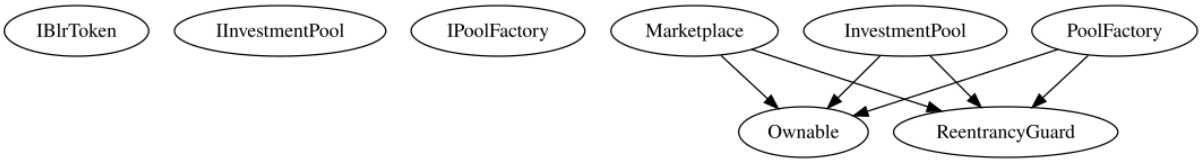
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IBlrToken	Interface			
	balanceOf	External		-
	allowance	External		-
	transferFrom	External	✓	-
	transfer	External	✓	-
IInvestmentPool	Interface			
	deposit	External	✓	-
	earlyWithdraw	External	✓	-
	withdrawAfterMaturity	External	✓	-
	modifyApprovedStablecoin	External	✓	-
	isPoolFullySubscribed	External		-
	totalDeposits	External		-
	poolActive	External		-
	poolEndTime	External		-
	investors	External		-
	withdrawAfterMaturityFor	External	✓	-
IPoolFactory	Interface			

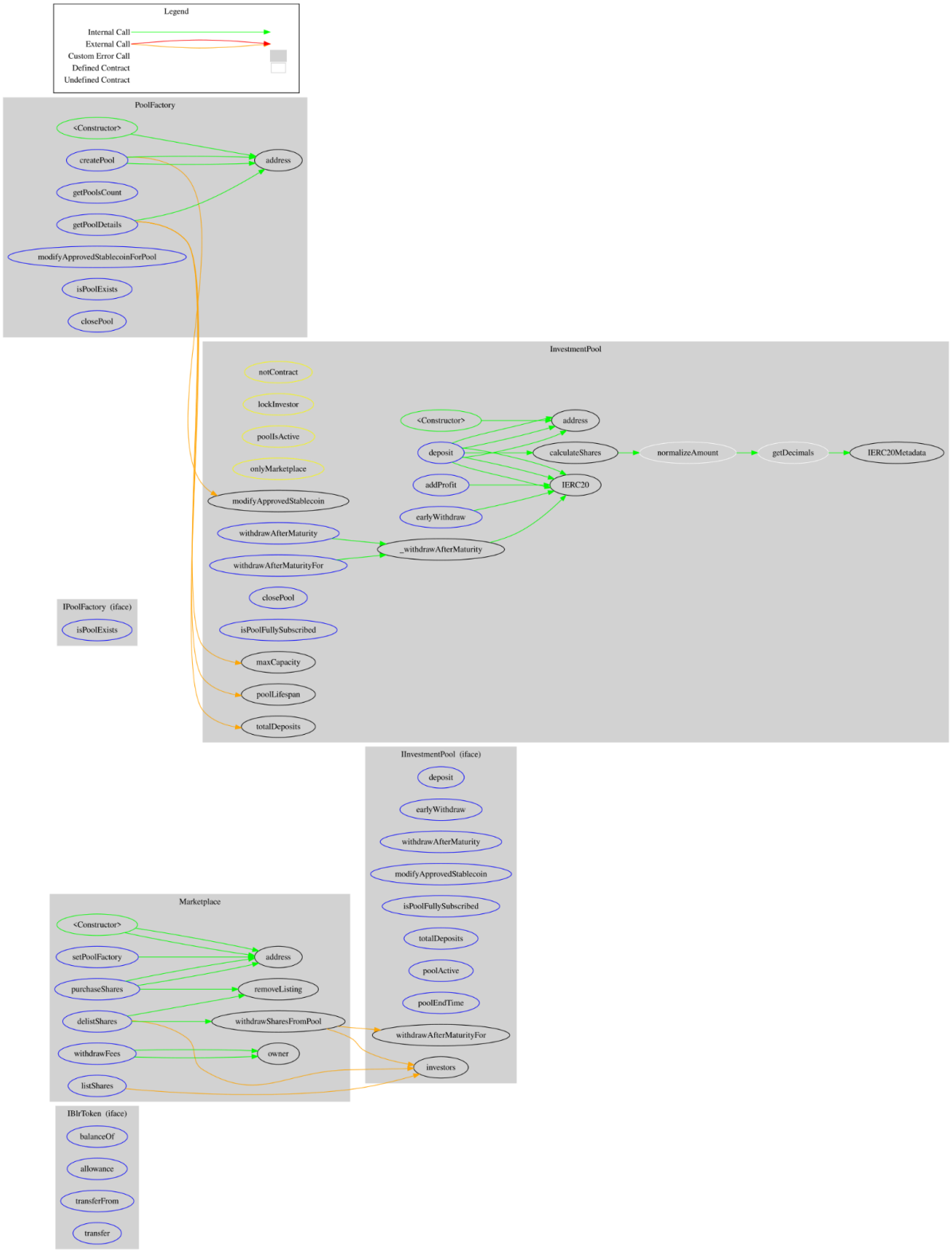
	isPoolExists	External		-
Marketplace	Implementation	Ownable, ReentrancyGuard		
		Public	✓	Ownable
	listShares	External	✓	nonReentrant
	purchaseShares	External	✓	nonReentrant
	delistShares	External	✓	nonReentrant
	withdrawSharesFromPool	Internal	✓	
	removeListing	Internal	✓	
	withdrawFees	External	✓	onlyOwner nonReentrant
	setPoolFactory	External	✓	onlyOwner
InvestmentPool	Implementation	ReentrancyGuard, Ownable		
		Public	✓	Ownable
	modifyApprovedStablecoin	External	✓	onlyOwner
	deposit	External	✓	poolIsActive nonReentrant notContract
	getDecimals	Internal		
	normalizeAmount	Internal		
	calculateShares	Internal		
	earlyWithdraw	External	✓	poolIsActive nonReentrant notContract lockInvestor

	withdrawAfterMaturity	External	✓	nonReentrant
	withdrawAfterMaturityFor	External	✓	onlyMarketplace nonReentrant
	_withdrawAfterMaturity	Internal	✓	notContract lockInvestor
	closePool	External	✓	onlyOwner
	addProfit	External	✓	onlyOwner nonReentrant
	isPoolFullySubscribed	External		-
PoolFactory	Implementation	Ownable, ReentrancyGuard		
		Public	✓	Ownable
	createPool	External	✓	onlyOwner nonReentrant
	getPoolsCount	External		-
	getPoolDetails	External		-
	modifyApprovedStablecoinForPool	External	✓	onlyOwner nonReentrant
	isPoolExists	External		-
	closePool	External	✓	onlyOwner nonReentrant

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Custom Error Call
Defined Contract
Undefined Contract
PoolFactory
<Constructor>
createPool
address
getPoolsCount
getPoolDetails
modifyApprovedStablecoinForPool
isPoolExists
closePool
InvestmentPool
notContract
lockInvestor
poolIsActive
onlyMarketplace
modifyApprovedStablecoin
withdrawAfterMaturity
withdrawAfterMaturityFor
closePool
isPoolFullySubscribed
maxCapacity
poolLifespan
totalDeposits
<Constructor>
address
deposit
calculateShares
normalizeAmount
getDecimals
IERC20Metadata
addProfit
IERC20
earlyWithdraw
_withdrawAfterMaturity
IPoolFactory (iface)
isPoolExists
IInvestmentPool (iface)
deposit
earlyWithdraw
withdrawAfterMaturity
modifyApprovedStablecoin
isPoolFullySubscribed
totalDeposits
poolActive
poolEndTime
withdrawAfterMaturityFor
investors
Marketplace
<Constructor>
setPoolFactory
address
purchaseShares
removeListing
delistShares
withdrawSharesFromPool
withdrawFees
owner
listShares
IBlrToken (iface)
balanceOf
allowance
transferFrom
transfer

Summary

BlaroThings 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

cyberscope.io