ARBICHECK

ShibZilla

Smart Contract Audit

arbicheck.com

arbicheck

Audited on May 09 2023

ARBICHECK

Table of Contents

ARBICHECK

Audit Summary

Project Name	ShibZilla
Contract Address	0x860A43f58771730596Ac0aa382e31f157752bA06
Deployer Address	0x4af7f86c70a6fba4ed9d49074d0805a3c63b1e5b
Website	https://shibazilla.finance/
Language	Solidity
Blockchain	Arbitrum
Audit Date	May 09 2023

The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

This audit report has been prepared by Arbicheck's experts at the request of the client. The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Audit Scope

Arbicheck team was commissioned by ShibZilla to perform an audit based on the following smart contracts:

https://arbiscan.io/address/0x860A43f58771730596Ac0aa382e31f157752bA06#code

Note that we **ONLY** audited the code available to us on this URL at the time of the audit. If the URL is not from any block explorer (main net), it may be subject to change. Always check the contract address on this audit report and compare it to the token you are doing research for.

ARBICHECK

SWC Attack Analysis

The Smart Contract Weakness Classification Registry (SWC Registry) is an implementation of the weakness classification scheme proposed in EIP-1470.

ID	Description	Status
SWC-100	Function Default Visibility	Not Found
SWC-101	Integer Overflow and Underflow	Not Found
SWC-102	Outdated Compiler Version	Not Found
SWC-103	Floating Pragma	Not Found
SWC-104	Unchecked Call Return Value	Not Found
SWC-105	Unprotected Ether Withdrawal	Not Found
SWC-106	Unprotected SELF DESTRUCT Instruction	Not Found
SWC-107	Reentrancy	Not Found
SWC-108	State Variable Default Visibility	Not Found
SWC-109	Uninitialized Storage Pointer	Not Found
SWC-110	Assert Violation	Not Found
SWC-111	Use of Deprecated Solidity Functions	Not Found
SWC-112	Delegatecall to Untrusted Callee	Not Found
SWC-113	DoS with Failed Call	Not Found
SWC-114	Transaction Order Dependence	Not Found
SWC-115	Authorization through tx.origin	Not Found
SWC-116	Block values as a proxy for time	Not Found

SWC-117	Signature Malleability	Not Found
SWC-118	Incorrect Constructor Name	Not Found
SWC-119	Shadowing State Variables	Not Found
SWC-120	Weak Sources of Randomness from Chain Attributes	Not Found
SWC-121	Missing Protection against Signature Replay Attacks	Not Found
SWC-122	Lack of Proper Signature Verification	Not Found
SWC-123	Requirement Violation	Not Found
SWC-124	Write to Arbitrary Storage Location	Not Found
SWC-125	Incorrect Inheritance Order	Not Found
SWC-126	Insufficient Gas Grieng	Not Found
SWC-127	Arbitrary Jump with Function Type Variable	Not Found
SWC-128	DoS With Block Gas Limit	Not Found
SWC-129	Typographical Error	Not Found
SWC-130	Right-To-Left-Override control character (U+202E)	Not Found
SWC-131	Presence of unused variables	Found
SWC-132	Unexpected Ether balance	Not Found
SWC-133	Hash Collisions With Multiple Variable Length Arguments	Not Found
SWC-134	Message call with hardcoded gas amount	Not Found
SWC-135	Code With No Effects	Not Found
SWC-136	Unencrypted Private Data On-Chain	Not Found

ARBICHECK

Slither Analysis

Impact	Confidence	Description
High	Medium	BABYTOKEN.addLiquidity(uint256,uint256) (a.sol#3300-3313) sends eth to arbitrary user Dangerous calls: - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0xdead),block.timestamp) (a.sol#3305-3312)
High	Medium	Reentrancy in BABYTOKEN._transfer(address,address,uint256) (a.sol#3139-3227): External calls: - swapAndSendToFee(marketingTokens) (a.sol#3170) - IERC20(rewardToken).transfer(_marketingWalletAddress,newBalance) (a.sol#3238) - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (a.sol#3291-3297) - swapAndLiquify(swapTokens) (a.sol#3177) - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0xdead),block.timestamp) (a.sol#3305-3312) - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (a.sol#3273-3279) - swapAndSendDividends(sellTokens) (a.sol#3182) - success = IERC20(rewardToken).transfer(address(dividendTracker),dividends) (a.sol#3318-3321) - dividendTracker.distributeCAKEDividends(dividends) (a.sol#3324) - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (a.sol#3291-3297) External calls sending eth: - swapAndLiquify(swapTokens) (a.sol#3177) - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0xdead),block.timestamp) (a.sol#3305-3312) State variables written after the call(s): - super._transfer(from,address(this),fees) (a.sol#3199) - _balances[sender] = senderBalance - amount (a.sol#379) - _balances[recipient] += amount (a.sol#381)

		ERC20._balances (a.sol#181) can be used in cross function reentrancies: - ERC20._mint(address,uint256) (a.sol#397-407) - ERC20._transfer(address,address,uint256) (a.sol#366-386) - ERC20.balanceOf(address) (a.sol#246-248) - super._transfer(from,to,amount) (a.sol#3202) - _balances[sender] = senderBalance - amount (a.sol#379) - _balances[recipient] += amount (a.sol#381) ERC20._balances (a.sol#181) can be used in cross function reentrancies: - ERC20._mint(address,uint256) (a.sol#397-407) - ERC20._transfer(address,address,uint256) (a.sol#366-386) - ERC20.balanceOf(address) (a.sol#246-248) - swapping = false (a.sol#3185) BABYTOKEN.swapping (a.sol#2814) can be used in cross function reentrancies: - BABYTOKEN._transfer(address,address,uint256) (a.sol#3139-3227)
High	Medium	BABYTOKEN.swapAndSendToFee(uint256) (a.sol#3229-3239) ignores return value by IERC20(rewardToken).transfer(_marketingWalletAddress,newBalance) (a.sol#3238)
Medium	Medium	Reentrancy in BABYTOKENDividendTracker.process(uint256) (a.sol#2692-2744): External calls: - processAccount(address(account),true) (a.sol#2725) - success = IERC20(rewardToken).transfer(user,_withdrawableDividend) (a.sol#2370-2373) State variables written after the call(s): - lastProcessedIndex = _lastProcessedIndex (a.sol#2741) BABYTOKENDividendTracker.lastProcessedIndex (a.sol#2503) can be used in cross function reentrancies: - BABYTOKENDividendTracker.getAccount(address) (a.sol#2595-2642) - BABYTOKENDividendTracker.getLastProcessedIndex() (a.sol#2587-2589) - BABYTOKENDividendTracker.lastProcessedIndex (a.sol#2503) - BABYTOKENDividendTracker.process(uint256) (a.sol#2692-2744)
Medium	Medium	Reentrancy in DividendPayingToken._withdrawDividendOfUser(address) (a.sol#2360-2386): External calls: - success = IERC20(rewardToken).transfer(user,_withdrawableDividend) (a.sol#2370-2373) State variables written after the call(s): - withdrawnDividends[user] = withdrawnDividends[user].sub(_withdrawableDividend) (a.sol#2376-2378)

		DividendPayingToken.withdrawnDividends (a.sol#2325) can be used in cross function reentrancies: - DividendPayingToken._withdrawDividendOfUser(address) (a.sol#2360-2386) - DividendPayingToken.withdrawableDividendOf(address) (a.sol#2398-2405) - DividendPayingToken.withdrawnDividendOf(address) (a.sol#2410-2417)
Medium	Medium	BABYTOKEN._transfer(address,address,uint256).iterations (a.sol#3213) is a local variable never initialized
Medium	Medium	BABYTOKEN._transfer(address,address,uint256).lastProcessedIndex (a.sol#3215) is a local variable never initialized
Medium	Medium	BABYTOKEN._transfer(address,address,uint256).claims (a.sol#3214) is a local variable never initialized
Medium	Medium	BABYTOKEN.addLiquidity(uint256,uint256) (a.sol#3300-3313) ignores return value by uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0xdead),block.timestamp) (a.sol#3305-3312)
Medium	Medium	BABYTOKEN._transfer(address,address,uint256) (a.sol#3139-3227) ignores return value by dividendTracker.process(gas) (a.sol#3212-3225)
Medium	Medium	BABYTOKEN.claim() (a.sol#3127-3129) ignores return value by dividendTracker.processAccount(address(msg.sender),false) (a.sol#3128)
Low	High	DividendPayingToken.__DividendPayingToken_init(address,string,string)._name (a.sol#2331) shadows: - ERC20Upgradeable._name (a.sol#1597) (state variable)
Low	High	DividendPayingToken.dividendOf(address)._owner (a.sol#2391) shadows: - OwnableUpgradeable._owner (a.sol#1941) (state variable)
Low	High	DividendPayingToken.withdrawnDividendOf(address)._owner (a.sol#2410) shadows: - OwnableUpgradeable._owner (a.sol#1941) (state variable)
Low	High	DividendPayingToken.accumulativeDividendOf(address)._owner (a.sol#2424) shadows: - OwnableUpgradeable._owner (a.sol#1941) (state variable)
Low	High	DividendPayingToken.__DividendPayingToken_init(address,string,string)._symbol (a.sol#2332) shadows: - ERC20Upgradeable._symbol (a.sol#1598) (state variable)
Low	High	DividendPayingToken.withdrawableDividendOf(address)._owner (a.sol#2398) shadows: - OwnableUpgradeable._owner (a.sol#1941) (state variable)
Low	Medium	BABYTOKEN.setSwapTokensAtAmount(uint256) (a.sol#2935-2941) should emit an event for: - swapTokensAtAmount = amount (a.sol#2940)

Low	Medium	BABYTOKEN.setMarketingFee(uint256) (a.sol#2985-2989) should emit an event for: - marketingFee = value (a.sol#2986) - totalFees = tokenRewardsFee.add(liquidityFee).add(marketingFee) (a.sol#2987)
Low	Medium	BABYTOKEN.setTokenRewardsFee(uint256) (a.sol#2973-2977) should emit an event for: - totalFees = tokenRewardsFee.add(liquidityFee).add(marketingFee) (a.sol#2975)
Low	Medium	BABYTOKEN.setLiquiditFee(uint256) (a.sol#2979-2983) should emit an event for: - liquidityFee = value (a.sol#2980) - totalFees = tokenRewardsFee.add(liquidityFee).add(marketingFee) (a.sol#2981)
Low	Medium	BABYTOKEN.constructor(string,string,uint256,address[4],uint256[3],uint256,address,uint256).serviceFeeReceiver_ (a.sol#2872) lacks a zero-check on : - address(serviceFeeReceiver_).transfer(serviceFee_) (a.sol#2930)
Low	Medium	BABYTOKEN.constructor(string,string,uint256,address[4],uint256[3],uint256,address,uint256)._uniswapV2Pair (a.sol#2906-2907) lacks a zero-check on : - uniswapV2Pair = _uniswapV2Pair (a.sol#2909)
Low	Medium	DividendPayingToken._withdrawDividendOfUser(address) (a.sol#2360-2386) has external calls inside a loop: success = IERC20(rewardToken).transfer(user,_withdrawableDividend) (a.sol#2370-2373)
Low	Medium	Reentrancy in BABYTOKEN._transfer(address,address,uint256) (a.sol#3139-3227): External calls: - swapAndSendToFee(marketingTokens) (a.sol#3170) - IERC20(rewardToken).transfer(_marketingWalletAddress,newBalance) (a.sol#3238) - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (a.sol#3291-3297) - swapAndLiquify(swapTokens) (a.sol#3177) - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0xdead),block.timestamp) (a.sol#3305-3312) - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (a.sol#3273-3279) External calls sending eth: - swapAndLiquify(swapTokens) (a.sol#3177) - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0xdead),block.timestamp) (a.sol#3305-3312) State variables written after the call(s): - swapAndLiquify(swapTokens) (a.sol#3177)

		- _allowances[owner][spender] = amount (a.sol#458)
Low	Medium	Reentrancy in BABYTOKENDividendTracker.processAccount(address,bool) (a.sol#2746-2760): External calls: - amount = _withdrawDividendOfUser(account) (a.sol#2751) - success = IERC20(rewardToken).transfer(user,_withdrawableDividend) (a.sol#2370-2373) State variables written after the call(s): - lastClaimTimes[account] = block.timestamp (a.sol#2754)
Low	Medium	Reentrancy in BABYTOKEN._transfer(address,address,uint256) (a.sol#3139-3227): External calls: - swapAndSendToFee(marketingTokens) (a.sol#3170) - IERC20(rewardToken).transfer(_marketingWalletAddress,newBalance) (a.sol#3238) - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (a.sol#3291-3297) - swapAndLiquify(swapTokens) (a.sol#3177) - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0xdead),block.timestamp) (a.sol#3305-3312) - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (a.sol#3273-3279) - swapAndSendDividends(sellTokens) (a.sol#3182) - success = IERC20(rewardToken).transfer(address(dividendTracker),dividends) (a.sol#3318-3321) - dividendTracker.distributeCAKEDividends(dividends) (a.sol#3324) - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (a.sol#3291-3297) External calls sending eth: - swapAndLiquify(swapTokens) (a.sol#3177) - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0xdead),block.timestamp) (a.sol#3305-3312) State variables written after the call(s): - swapAndSendDividends(sellTokens) (a.sol#3182) - _allowances[owner][spender] = amount (a.sol#458)

Low	Medium	Reentrancy in BABYTOKEN.swapAndLiquify(uint256) (a.sol#3241-3262): External calls: - swapTokensForEth(half) (a.sol#3253) - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (a.sol#3273-3279) - addLiquidity(otherHalf,newBalance) (a.sol#3259) - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0xdead),block.timestamp) (a.sol#3305-3312) External calls sending eth: - addLiquidity(otherHalf,newBalance) (a.sol#3259) - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0xdead),block.timestamp) (a.sol#3305-3312) State variables written after the call(s): - addLiquidity(otherHalf,newBalance) (a.sol#3259) - _allowances[owner][spender] = amount (a.sol#458)
Low	Medium	Reentrancy in BABYTOKEN.processDividendTracker(uint256) (a.sol#3111-3125): External calls: - (iterations,claims,lastProcessedIndex) = dividendTracker.process(gas) (a.sol#3112-3116) Event emitted after the call(s): - ProcessedDividendTracker(iterations,claims,lastProcessedIndex,false,gas,tx.origin) (a.sol#3117-3124)
Low	Medium	Reentrancy in BABYTOKEN.swapAndSendDividends(uint256) (a.sol#3315-3327): External calls: - swapTokensForCake(tokens) (a.sol#3316) - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (a.sol#3291-3297) - success = IERC20(rewardToken).transfer(address(dividendTracker),dividends) (a.sol#3318-3321) - dividendTracker.distributeCAKEDividends(dividends) (a.sol#3324) Event emitted after the call(s): - SendDividends(tokens,dividends) (a.sol#3325)

Low	Medium	Reentrancy in BABYTOKEN._transfer(address,address,uint256) (a.sol#3139-3227): External calls: - swapAndSendToFee(marketingTokens) (a.sol#3170) - IERC20(rewardToken).transfer(_marketingWalletAddress,newBalance) (a.sol#3238) - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (a.sol#3291-3297) - swapAndLiquify(swapTokens) (a.sol#3177) - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0xdead),block.timestamp) (a.sol#3305-3312) - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (a.sol#3273-3279) External calls sending eth: - swapAndLiquify(swapTokens) (a.sol#3177) - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0xdead),block.timestamp) (a.sol#3305-3312) Event emitted after the call(s): - Approval(owner,spender,amount) (a.sol#459) - swapAndLiquify(swapTokens) (a.sol#3177) - SwapAndLiquify(half,newBalance,otherHalf) (a.sol#3261) - swapAndLiquify(swapTokens) (a.sol#3177)
Low	Medium	Reentrancy in BABYTOKEN.swapAndLiquify(uint256) (a.sol#3241-3262): External calls: - swapTokensForEth(half) (a.sol#3253) - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (a.sol#3273-3279) - addLiquidity(otherHalf,newBalance) (a.sol#3259) - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0xdead),block.timestamp) (a.sol#3305-3312) External calls sending eth: - addLiquidity(otherHalf,newBalance) (a.sol#3259) - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0xdead),block.timestamp) (a.sol#3305-3312) Event emitted after the call(s):

		- Approval(owner,spender,amount) (a.sol#459) - addLiquidity(otherHalf,newBalance) (a.sol#3259) - SwapAndLiquify(half,newBalance,otherHalf) (a.sol#3261)
Low	Medium	Reentrancy in BABYTOKENDividendTracker.processAccount(address,bool) (a.sol#2746-2760): External calls: - amount = _withdrawDividendOfUser(account) (a.sol#2751) - success = IERC20(rewardToken).transfer(user,_withdrawableDividend) (a.sol#2370-2373) Event emitted after the call(s): - Claim(account,amount,automatic) (a.sol#2755)
Low	Medium	Reentrancy in BABYTOKEN._setAutomatedMarketMakerPair(address,bool) (a.sol#2991-3003): External calls: - dividendTracker.excludeFromDividends(pair) (a.sol#2999) Event emitted after the call(s): - SetAutomatedMarketMakerPair(pair,value) (a.sol#3002)
Low	Medium	Reentrancy in BABYTOKEN._transfer(address,address,uint256) (a.sol#3139-3227): External calls: - swapAndSendToFee(marketingTokens) (a.sol#3170) - IERC20(rewardToken).transfer(_marketingWalletAddress,newBalance) (a.sol#3238) - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (a.sol#3291-3297) - swapAndLiquify(swapTokens) (a.sol#3177) - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0xdead),block.timestamp) (a.sol#3305-3312) - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (a.sol#3273-3279) - swapAndSendDividends(sellTokens) (a.sol#3182) - success = IERC20(rewardToken).transfer(address(dividendTracker),dividends) (a.sol#3318-3321) - dividendTracker.distributeCAKEDividends(dividends) (a.sol#3324) - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (a.sol#3291-3297)

		External calls sending eth: - swapAndLiquify(swapTokens) (a.sol#3177) - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0xdead),block.timestamp) (a.sol#3305-3312) Event emitted after the call(s): - Approval(owner,spender,amount) (a.sol#459) - swapAndSendDividends(sellTokens) (a.sol#3182) - SendDividends(tokens,dividends) (a.sol#3325) - swapAndSendDividends(sellTokens) (a.sol#3182) - Transfer(sender,recipient,amount) (a.sol#383) - super._transfer(from,address(this),fees) (a.sol#3199) - Transfer(sender,recipient,amount) (a.sol#383) - super._transfer(from,to,amount) (a.sol#3202)
Low	Medium	Reentrancy in BABYTOKEN._transfer(address,address,uint256) (a.sol#3139-3227): External calls: - swapAndSendToFee(marketingTokens) (a.sol#3170) - IERC20(rewardToken).transfer(_marketingWalletAddress,newBalance) (a.sol#3238) - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (a.sol#3291-3297) - swapAndLiquify(swapTokens) (a.sol#3177) - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0xdead),block.timestamp) (a.sol#3305-3312) - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (a.sol#3273-3279) - swapAndSendDividends(sellTokens) (a.sol#3182) - success = IERC20(rewardToken).transfer(address(dividendTracker),dividends) (a.sol#3318-3321) - dividendTracker.distributeCAKEDividends(dividends) (a.sol#3324) - uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (a.sol#3291-3297) - dividendTracker.setBalance(address(from),balanceOf(from)) (a.sol#3204-3206) - dividendTracker.setBalance(address(to),balanceOf(to)) (a.sol#3207) - dividendTracker.process(gas) (a.sol#3212-3225)

		External calls sending eth: - swapAndLiquify(swapTokens) (a.sol#3177) - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,address(0xdead),block.timestamp) (a.sol#3305-3312) Event emitted after the call(s): - ProcessedDividendTracker(iterations,claims,lastProcessedIndex,true,gas,tx.origin) (a.sol#3217-3224)
Low	Medium	BABYTOKENDividendTracker.getAccount(address) (a.sol#2595-2642) uses timestamp for comparisons Dangerous comparisons: - nextClaimTime > block.timestamp (a.sol#2639-2641)
Low	Medium	BABYTOKENDividendTracker.canAutoClaim(uint256) (a.sol#2667-2673) uses timestamp for comparisons Dangerous comparisons: - lastClaimTime > block.timestamp (a.sol#2668) - block.timestamp.sub(lastClaimTime) >= claimWait (a.sol#2672)
Informational	High	Clones.cloneDeterministic(address,bytes32) (a.sol#849-858) uses assembly - INLINE ASM (a.sol#850-856)
Informational	High	Clones.clone(address) (a.sol#831-840) uses assembly - INLINE ASM (a.sol#832-838)
Informational	High	Clones.predictDeterministicAddress(address,bytes32,address) (a.sol#863-878) uses assembly - INLINE ASM (a.sol#868-877)
Informational	High	Address.verifyCallResult(bool,bytes,string) (a.sol#1088-1108) uses assembly - INLINE ASM (a.sol#1100-1103)
Informational	High	Address.isContract(address) (a.sol#919-929) uses assembly - INLINE ASM (a.sol#925-927)
Informational	High	BABYTOKEN._transfer(address,address,uint256) (a.sol#3139-3227) has a high cyclomatic complexity (14).
Informational	Medium	ContextUpgradeable._msgData() (a.sol#1548-1550) is never used and should be removed
Informational	Medium	Address.verifyCallResult(bool,bytes,string) (a.sol#1088-1108) is never used and should be removed
Informational	Medium	SafeMathInt.div(int256,int256) (a.sol#2083-2089) is never used and should be removed
Informational	Medium	SafeMath.tryDiv(uint256,uint256) (a.sol#641-646) is never used and should be removed

Informational	Medium	Address.sendValue(address,uint256) (a.sol#947-952) is never used and should be removed
Informational	Medium	Address.functionCallWithValue(address,bytes,uint256) (a.sol#1001-1007) is never used and should be removed
Informational	Medium	Clones.predictDeterministicAddress(address,bytes32) (a.sol#883-889) is never used and should be removed
Informational	Medium	SafeMathInt.abs(int256) (a.sol#2112-2115) is never used and should be removed
Informational	Medium	SafeMath.tryMod(uint256,uint256) (a.sol#653-658) is never used and should be removed
Informational	Medium	Address.functionDelegateCall(address,bytes,string) (a.sol#1071-1080) is never used and should be removed
Informational	Medium	SafeMath.sub(uint256,uint256,string) (a.sol#745-754) is never used and should be removed
Informational	Medium	SafeMathInt.mul(int256,int256) (a.sol#2071-2078) is never used and should be removed
Informational	Medium	Clones.predictDeterministicAddress(address,bytes32,address) (a.sol#863-878) is never used and should be removed
Informational	Medium	Address.functionDelegateCall(address,bytes) (a.sol#1061-1063) is never used and should be removed
Informational	Medium	SafeMath.tryAdd(uint256,uint256) (a.sol#599-605) is never used and should be removed
Informational	Medium	ERC20._burn(address,uint256) (a.sol#420-435) is never used and should be removed
Informational	Medium	SafeMath.mod(uint256,uint256,string) (a.sol#794-803) is never used and should be removed
Informational	Medium	Address.functionCallWithValue(address,bytes,uint256,string) (a.sol#1015-1026) is never used and should be removed
Informational	Medium	SafeMath.div(uint256,uint256,string) (a.sol#768-777) is never used and should be removed
Informational	Medium	Context._msgData() (a.sol#140-142) is never used and should be removed
Informational	Medium	Address.functionStaticCall(address,bytes) (a.sol#1034-1036) is never used and should be removed

Informational	Medium	ERC20Upgradeable._transfer(address,address,uint256) (a.sol#1781-1801) is never used and should be removed
Informational	Medium	SafeMath.mod(uint256,uint256) (a.sol#728-730) is never used and should be removed
Informational	Medium	Clones.cloneDeterministic(address,bytes32) (a.sol#849-858) is never used and should be removed
Informational	Medium	SafeMath.tryMul(uint256,uint256) (a.sol#624-634) is never used and should be removed
Informational	Medium	SafeMath.trySub(uint256,uint256) (a.sol#612-617) is never used and should be removed
Informational	Medium	DividendPayingToken._transfer(address,address,uint256) (a.sol#2443-2458) is never used and should be removed
Informational	Medium	Address.functionCall(address,bytes,string) (a.sol#982-988) is never used and should be removed
Informational	Medium	Address.functionStaticCall(address,bytes,string) (a.sol#1044-1053) is never used and should be removed
Informational	Medium	ContextUpgradeable.__Context_init() (a.sol#1538-1540) is never used and should be removed
Informational	Medium	Address.functionCall(address,bytes) (a.sol#972-974) is never used and should be removed
Informational	High	Pragma version=0.8.4 (a.sol#2791) allows old versions
Informational	High	solc-0.8.4 is not recommended for deployment
Informational	High	Low level call in Address.functionStaticCall(address,bytes,string) (a.sol#1044-1053): - (success,returndata) = target.staticcall(data) (a.sol#1051)
Informational	High	Low level call in Address.functionDelegateCall(address,bytes,string) (a.sol#1071-1080): - (success,returndata) = target.delegatecall(data) (a.sol#1078)
Informational	High	Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (a.sol#1015-1026): - (success,returndata) = target.call{value: value}(data) (a.sol#1024)
Informational	High	Low level call in Address.sendValue(address,uint256) (a.sol#947-952): - (success) = recipient.call{value: amount}() (a.sol#950)
Informational	High	Parameter DividendPayingToken.__DividendPayingToken_init(address,string,string)._name (a.sol#2331) is not in mixedCase

Informational	High	Variable ContextUpgradeable.__gap (a.sol#1551) is not in mixedCase
Informational	High	Parameter BABYTOKENDividendTracker.getAccount(address)._account (a.sol#2595) is not in mixedCase
Informational	High	Function IUniswapV2Pair.PERMIT_TYPEHASH() (a.sol#2021) is not in mixedCase
Informational	High	Function OwnableUpgradeable.__Ownable_init() (a.sol#1948-1951) is not in mixedCase
Informational	High	Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (a.sol#2038) is not in mixedCase
Informational	High	Parameter DividendPayingToken.__DividendPayingToken_init(address,string,string)._rewardToken (a.sol#2330) is not in mixedCase
Informational	High	Constant DividendPayingToken.magnitude (a.sol#2309) is not in UPPER_CASE_WITH_UNDERSCORES
Informational	High	Variable OwnableUpgradeable.__gap (a.sol#1997) is not in mixedCase
Informational	High	Function IUniswapV2Pair.DOMAIN_SEPARATOR() (a.sol#2020) is not in mixedCase
Informational	High	Function ERC20Upgradeable.__ERC20_init_unchained(string,string) (a.sol#1614-1617) is not in mixedCase
Informational	High	Function ERC20Upgradeable.__ERC20_init(string,string) (a.sol#1609-1612) is not in mixedCase
Informational	High	Function ContextUpgradeable.__Context_init_unchained() (a.sol#1542-1543) is not in mixedCase
Informational	High	Variable ERC20Upgradeable.__gap (a.sol#1916) is not in mixedCase
Informational	High	Parameter DividendPayingToken.dividendOf(address)._owner (a.sol#2391) is not in mixedCase
Informational	High	Parameter DividendPayingToken.withdrawnDividendOf(address)._owner (a.sol#2410) is not in mixedCase
Informational	High	Variable BABYTOKEN._marketingWalletAddress (a.sol#2827) is not in mixedCase
Informational	High	Parameter DividendPayingToken.__DividendPayingToken_init(address,string,string)._symbol (a.sol#2332) is not in mixedCase

Informational	High	Parameter DividendPayingToken.withdrawableDividendOf(address)._owner (a.sol#2398) is not in mixedCase
Informational	High	Function OwnableUpgradeable.__Ownable_init_unchained() (a.sol#1953-1955) is not in mixedCase
Informational	High	Function IUniswapV2Router01.WETH() (a.sol#1154) is not in mixedCase
Informational	High	Parameter DividendPayingToken.accumulativeDividendOf(address)._owner (a.sol#2424) is not in mixedCase
Informational	High	Function ContextUpgradeable.__Context_init() (a.sol#1538-1540) is not in mixedCase
Informational	High	Function DividendPayingToken.__DividendPayingToken_init(address,string,string) (a.sol#2329-2337) is not in mixedCase
Informational	Medium	Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (a.sol#1159) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (a.sol#1160)
Informational	Medium	Variable DividendPayingToken.__DividendPayingToken_init(address,string,string)._rewardToken (a.sol#2330) is too similar to BABYTOKENDividendTracker.initialize(address,uint256).rewardToken_ (a.sol#2522)
Informational	Medium	Variable DividendPayingToken._withdrawDividendOfUser(address)._withdrawableDividend (a.sol#2364) is too similar to BABYTOKENDividendTracker.getAccount(address).withdrawableDividends (a.sol#2602)
Informational	Medium	BABYTOKEN.constructor(string,string,uint256,address[4],uint256[3],uint256,address,uint256) (a.sol#2865-2931) uses literals with too many digits: - gasForProcessing = 300000 (a.sol#2894)
Informational	Medium	Clones.cloneDeterministic(address,bytes32) (a.sol#849-858) uses literals with too many digits: - mstore(uint256,uint256)(ptr_cloneDeterministic_asm_0 + 0x28,0x5af43d82803e903d91602b57fd5bf3000000000000000000000000000000000000) (a.sol#854)
Informational	Medium	Clones.predictDeterministicAddress(address,bytes32,address) (a.sol#863-878) uses literals with too many digits: - mstore(uint256,uint256)(ptr_predictDeterministicAddress_asm_0 + 0x28,0x5af43d82803e903d91602b57fd5bf3ff0000000000000000000000000000000000) (a.sol#872)

Informational	Medium	Clones.clone(address) (a.sol#831-840) uses literals with too many digits: - mstore(uint256,uint256)(ptr_clone_asm_0 + 0x28,0x5af43d82803e903d91602b57fd5bf3000000000000000000000000000000000000) (a.sol#836)
Informational	Medium	Clones.predictDeterministicAddress(address,bytes32,address) (a.sol#863-878) uses literals with too many digits: - mstore(uint256,uint256)(ptr_predictDeterministicAddress_asm_0,0x3d602d80600a3d3981f3363d3d373d3d3d363d7300000000000000000000000000) (a.sol#870)
Informational	Medium	Clones.cloneDeterministic(address,bytes32) (a.sol#849-858) uses literals with too many digits: - mstore(uint256,uint256)(ptr_cloneDeterministic_asm_0,0x3d602d80600a3d3981f3363d3d373d3d3d363d7300000000000000000000000000) (a.sol#852)
Informational	Medium	BABYTOKEN.updateGasForProcessing(uint256) (a.sol#3005-3016) uses literals with too many digits: - require(bool,string)(newValue >= 200000 && newValue <= 500000,BABYTOKEN: gasForProcessing must be between 200,000 and 500,000) (a.sol#3006-3009)
Informational	Medium	Clones.clone(address) (a.sol#831-840) uses literals with too many digits: - mstore(uint256,uint256)(ptr_clone_asm_0,0x3d602d80600a3d3981f3363d3d373d3d3d363d7300000000000000000000) (a.sol#834)
Informational	High	SafeMathInt.MAX_INT256 (a.sol#2066) is never used in SafeMathInt (a.sol#2064-2121)
Optimization	High	BABYTOKEN.dividendTracker (a.sol#2816) should be immutable
Optimization	High	BABYTOKEN.uniswapV2Router (a.sol#2811) should be immutable
Optimization	High	BABYTOKEN.rewardToken (a.sol#2818) should be immutable
Optimization	High	BABYTOKEN.uniswapV2Pair (a.sol#2812) should be immutable

Risk Classification

Arbicheck uses certain vulnerability levels, these indicate how bad a certain issue is. The higher the risk, the more strictly it is recommended to correct the error before using the contract.

Vulnerability Level	Description
High Issues	These issues will cause the problems and SHOULD be adjusted
Medium Issues	These issues will likely cause the problems and recommended to be adjusted
Low Issues	These issues will not cause any problems, but can be adjusted for the improvement
Notes	Does not compromise the functionality of the contract and just the general recommendations

Succeeded Transfer check

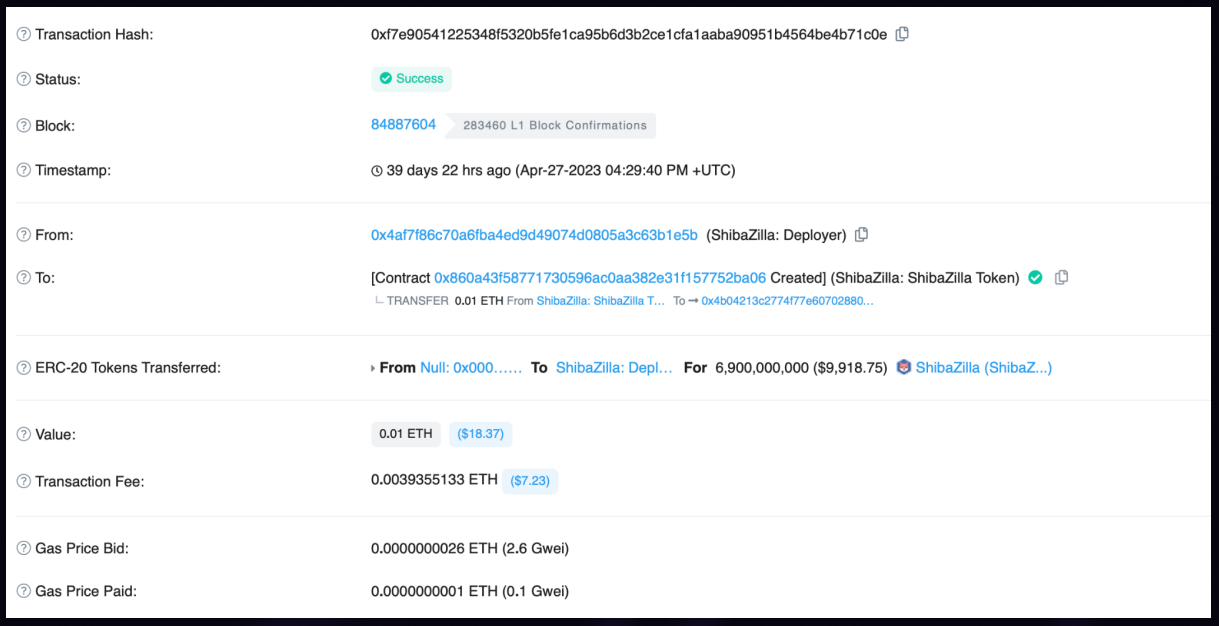
Low Issue

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

ARBICHECK

Deploy Snapshot



Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Arbicheck and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Arbicheck) owe no duty of care towards you or any other person, nor does Arbicheck make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties, or other terms of any kind except as set out in this disclaimer, and Arbicheck hereby excludes all representations, warranties, conditions, and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Arbicheck hereby excludes all liability and

responsibility, and neither you nor any other person shall have any claim against Arbicheck, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of the use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed. Arbicheck owns all copyright rights to the text, images, photographs, and other content provided in the following document. When using or sharing partly or in full, third parties must provide a direct link to the original document mentioning the author (arbicheck.com).

ARBICHECK