RD AUDITORS

Blokkpay Contract Code Review and Security Analysis Report

Customer: Blokkpay
Prepared on: 12th April 2024
Platform: Ethereum
Language: Solidity

rdauditors.com

Table of Contents

Disclaimer

This document may contain confidential information about its systems and intellectual property of the customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the customer or it can be disclosed publicly after all vulnerabilities are fixed - upon the decision of the customer.

Document

Name	Smart Contract Code Review and Security Analysis Report of Blokkpay
Platform	Ethereum/ Solidity
File 1	Escrow.sol
MD5 hash	fdd7025ceaeee696e9d5129278c30529
SHA256 hash	e0a35eaa2fb83415a11927254ca0986b503819ef245b28eaa1f3e63ffa458ab4
File 2	BlockToken..sol
MD5 hash	982bc7dddc5589d7edbc8f5123dc7a6a
SHA256 hash	d607d8bac28320f4ef3831e415037b1fea3bceabb3dcc58be228598e267eac1e
Date	12/04/2024

Introduction

RD Auditors (Consultant) were contracted by Blokkpay (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report represents the findings of the security assessment of the customer's smart contract and its code review conducted between 6th - 12th April 2024.

This contract consists of two files.

Project Scope

The scope of the project is a smart contract. We have scanned this smart contract for commonly known and more specific vulnerabilities, below are those considered (the full list includes but is not limited to):

- Reentrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Byte array vulnerabilities
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Unchecked external call - Unchecked math
- Unsafe type inference
- Implicit visibility level

Executive Summary

According to the assessment, the customer's solidity smart contract is now **Well-Secured.**

Automated checks are with smartDec, Mythril, Slither and remix IDE. All issues were performed by our team, which included the analysis of code functionality, the manual audit found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the audit overview section. The general overview is presented in the AS-IS section and all issues found are located in the audit overview section.

We found the following;

Total Issues	0
Critical	0
High	0
Medium	0
Low	0
Very Low	0

Code Quality

The libraries within this smart contract are part of a logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned to a specific address and its properties/methods can be reused many times by other contracts.

The Blokkpay team has provided scenario and unit test scripts, which would help to determine the integrity of the code in an automated way.

Overall, the code is almost commented. Commenting can provide rich documentation for functions, return variables and more. Use of the Ethereum Natural Language Specification Format (NatSpec) for commenting is recommended.

Documentation

We were given the Blokkpay code as a zip file.

The hash of that file is mentioned in the table. As mentioned above, it's commented on smart contract code, so anyone can quickly understand the programming flow as well as complex code logic.

Comments are very helpful in understanding the overall architecture of the protocol. It also provides a clear overview of the system components, including helpful details, like the lifetime of the background script.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure. Those were based on well known industry standard open source projects and even core code blocks that are written well and systematically.

AS-IS Overview

Escrow.sol

File And Function Level Report

Contract: MarketplaceEscrowV1

Inherit: Ownable, ReentrancyGuard

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	depositBNB	external	Passed	All Passed	No Issue	Passed
2	depositBEP20	external	Passed	All Passed	No Issue	Passed
3	_createJob	internal	Passed	All Passed	No Issue	Passed
4	_createMilestone	internal	Passed	All Passed	No Issue	Passed
5	releaseAsClient	external	Passed	All Passed	No Issue	Passed
6	releaseByProvider	external	Passed	All Passed	No Issue	Passed
7	releaseByAdmin	external	Passed	All Passed	No Issue	Passed
8	withdrawQueuedPayment	external	Passed	All Passed	No Issue	Passed
9	_release	internal	Passed	All Passed	No Issue	Passed

10	_checkDiscountEligiblity	internal	Passed	All Passed	No Issue	Passed
11	_calculateDiscount	internal	Passed	All Passed	No Issue	Passed
12	raiseDispute	external	Passed	All Passed	No Issue	Passed
13	_pancakeSwapTokens	internal	Passed	All Passed	No Issue	Passed
14	_pancakeSwapBNB	internal	Passed	All Passed	No Issue	Passed
15	withdrawRevenue	external	Passed	All Passed	No Issue	Passed
16	changeDefaultFee	external	Passed	All Passed	No Issue	Passed
17	changeDefaultDiscount	external	Passed	All Passed	No Issue	Passed
18	changeSwapRouter	external	Passed	All Passed	No Issue	Passed
19	viewDefaultFee	external	Passed	All Passed	No Issue	Passed
20	viewDefaultDiscountPercent	external	Passed	All Passed	No Issue	Passed
21	viewMaxMilestonePerJob	external	Passed	All Passed	No Issue	Passed
22	viewDiscountUSDT	external	Passed	All Passed	No Issue	Passed
23	isMilestoneEnabled	external	Passed	All Passed	No Issue	Passed
24	getJobs	external	Passed	All Passed	No Issue	Passed

BlockToken.sol

Contract: BlockToken

Inherit: Ownable

Observation: Passed

Test Report: Passed

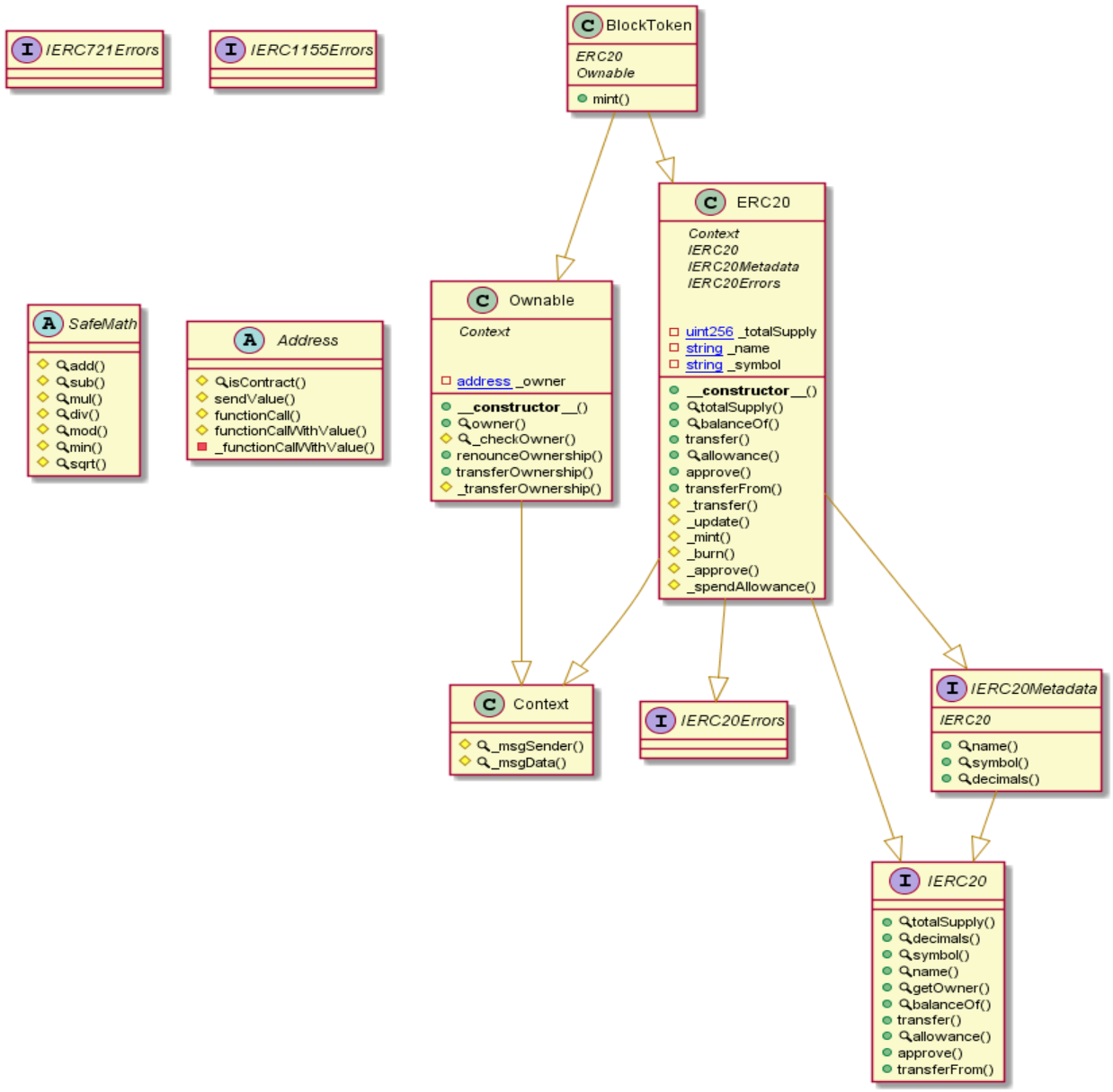
Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	mint	public	Passed	All Passed	No Issue	Passed

Code Flow Diagram - BlockToken

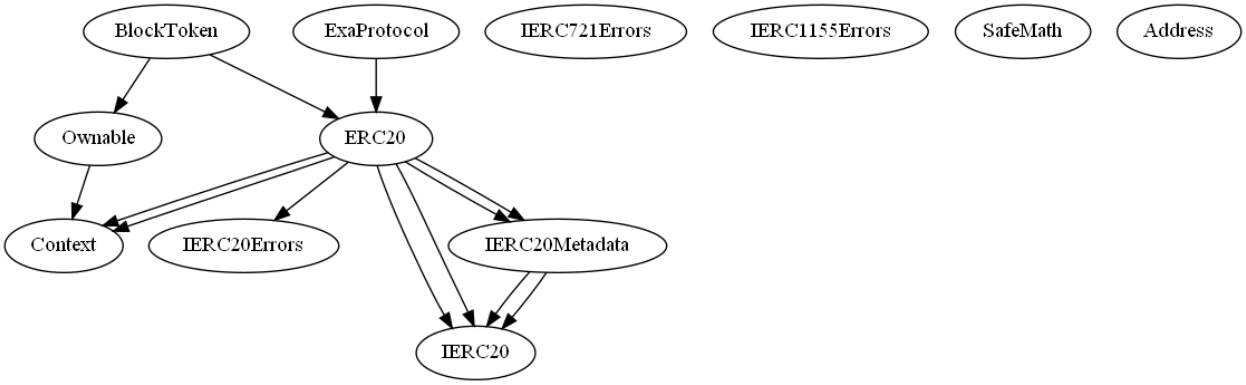
Please follow the below link to see the image clearly:

https://drive.google.com/file/d/1SytKW8L3nZNRn1JUJg7tsLfOc16Xv8TF/view?usp=drive_link

UML Diagram - BlockToken



Inheritance Diagram - BlockToken

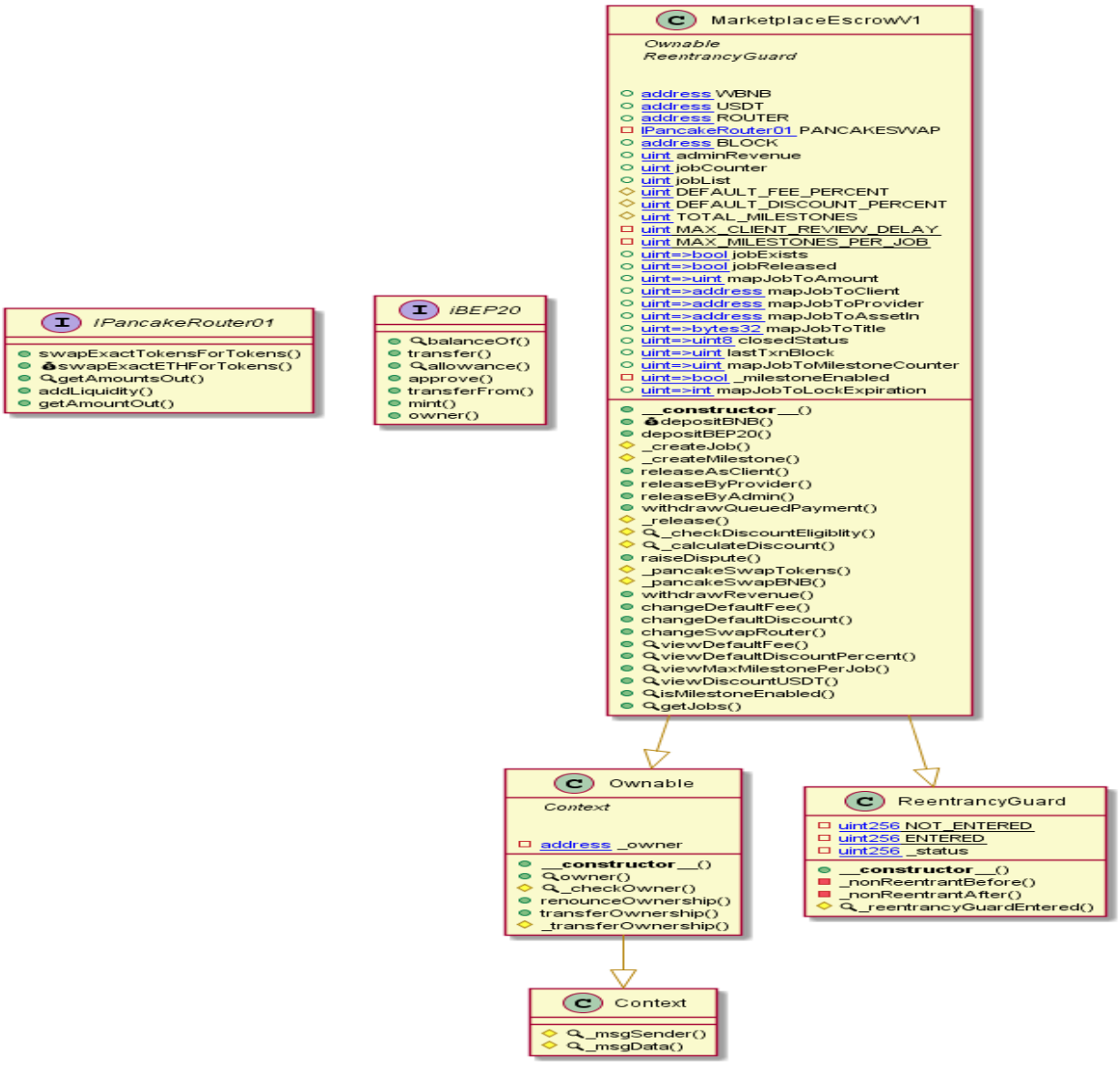


Code Flow Diagram - Escrow

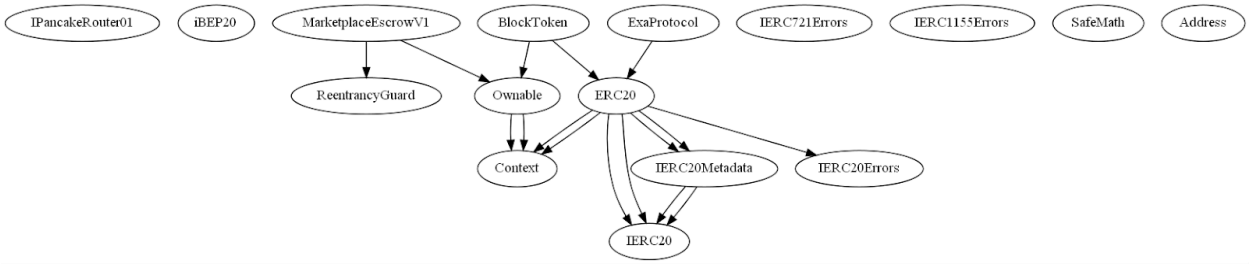
Please follow the below link to see the image clearly:

https://drive.google.com/file/d/1L11NMfUhD8GkR7Buac5zy4Bdpf_g_-Ry/view?usp=drive_link

UML Diagram - Escrow



Inheritance Diagram - Escrow



Code Flow Diagram - Slither Results Log

BlockToken.sol

```
INFO:Detectors:
Address.isContract(address) (BlockToken.sol#99-106) uses assembly
        - INLINE ASM (BlockToken.sol#102-104)
Address._functionCallWithValue(address,bytes,uint256,string) (BlockToken.sol#145-167) uses assemb
ly
        - INLINE ASM (BlockToken.sol#159-162)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (BlockToken.sol#145-167) is never us
ed and should be removed
Address.functionCall(address,bytes) (BlockToken.sol#115-117) is never used and should be removed
Address.functionCall(address,bytes,string) (BlockToken.sol#119-125) is never used and should be r
emoved
Address.functionCallWithValue(address,bytes,uint256) (BlockToken.sol#127-133) is never used and s
hould be removed
Address.functionCallWithValue(address,bytes,uint256,string) (BlockToken.sol#135-143) is never use
d and should be removed
Address.isContract(address) (BlockToken.sol#99-106) is never used and should be removed
Address.sendValue(address,uint256) (BlockToken.sol#108-113) is never used and should be removed
Context._msgData() (BlockToken.sol#204-207) is never used and should be removed
ERC20._burn(address,uint256) (BlockToken.sol#481-486) is never used and should be removed
SafeMath.add(uint256,uint256) (BlockToken.sol#20-25) is never used and should be removed
SafeMath.div(uint256,uint256) (BlockToken.sol#53-55) is never used and should be removed
SafeMath.div(uint256,uint256,string) (BlockToken.sol#57-66) is never used and should be removed
SafeMath.min(uint256,uint256) (BlockToken.sol#81-83) is never used and should be removed
SafeMath.mod(uint256,uint256) (BlockToken.sol#68-70) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (BlockToken.sol#72-79) is never used and should be removed
SafeMath.mul(uint256,uint256) (BlockToken.sol#42-51) is never used and should be removed
SafeMath.sqrt(uint256) (BlockToken.sol#85-96) is never used and should be removed
```

```
SafeMath.sqrt(uint256) (BlockToken.sol#85-96) is never used and should be removed
SafeMath.sub(uint256,uint256) (BlockToken.sol#27-29) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (BlockToken.sol#31-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.7.5 (BlockToken.sol#3) allows old versions
solc-0.7.5 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-so
lidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (BlockToken.sol#108-113):
        - (success) = recipient.call{value: amount}() (BlockToken.sol#111)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (BlockToken.sol#14
5-167):
        - (success,returndata) = target.call{value: weiValue}(data) (BlockToken.sol#153)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter BlockToken.mint(address,uint256)._to (BlockToken.sol#559) is not in mixedCase
Parameter BlockToken.mint(address,uint256)._amount (BlockToken.sol#559) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-
naming-conventions
```

```
INFO:Detectors:
Redundant expression "this (BlockToken.sol#205)" inContext (BlockToken.sol#199-208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
BlockToken (BlockToken.sol#556-562) does not implement functions:
        - IERC20Metadata.decimals() (BlockToken.sol#306)
        - IERC20.getOwner() (BlockToken.sol#178)
        - IERC20Metadata.name() (BlockToken.sol#296)
        - IERC20Metadata.symbol() (BlockToken.sol#301)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Slither:BlockToken.sol analyzed (11 contracts with 93 detectors), 29 result(s) found
```

Escrow.sol

```
INFO:Detectors:
MarketplaceEscrowV1._release(uint256,address,uint256,bool)._finalRelease (escrow.sol#632) is a lo
cal variable never initialized
MarketplaceEscrowV1.getJobs(uint256,uint256).j (escrow.sol#851) is a local variable never initial
ized
MarketplaceEscrowV1.releaseByAdmin(uint256,uint256,uint256)._clientRelease (escrow.sol#579) is a
local variable never initialized
MarketplaceEscrowV1.releaseByAdmin(uint256,uint256,uint256)._provRelease (escrow.sol#583) is a lo
cal variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-vari
ables
INFO:Detectors:
iBEP20.allowance(address,address).owner (escrow.sol#55) shadows:
        - iBEP20.owner() (escrow.sol#65) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

INFO:Detectors:
Reentrancy in MarketplaceEscrowV1.depositBEP20(address,address,uint256,uint256,bytes32,address[],
bool) (escrow.sol#363-418):
        External calls:
        - require(bool,string)(iBEP20(asset).transferFrom(msg.sender,address(this),value),Marketp
laceEscrowV1: BEP20_TRANSFER_FAILED) (escrow.sol#376-379)
        State variables written after the call(s):
        - lastTxnBlock[JOBID] = block.number (escrow.sol#394)
        - mapJobToAssetIn[JOBID] = asset (escrow.sol#390)
        - mapJobToClient[JOBID] = msg.sender (escrow.sol#388)
        - mapJobToProvider[JOBID] = provider (escrow.sol#389)
        - mapJobToTitle[JOBID] = jobTitle (escrow.sol#391)
```

```
Reentrancy in MarketplaceEscrowV1.depositBEP20(address,address,uint256,uint256,bytes32,address[],
bool) (escrow.sol#363-418):
        External calls:
        - require(bool,string)(iBEP20(asset).transferFrom(msg.sender,address(this),value),Marketp
laceEscrowV1: BEP20_TRANSFER_FAILED) (escrow.sol#376-379)
        - require(bool,string)(iBEP20(asset).approve(ROUTER,value),Swap Router Approval Failure)
(escrow.sol#400-403)
        - _finalUSD = _pancakeSwapTokens(value,swapPath) (escrow.sol#404)
                - _amounts = PANCAKESWAP.swapExactTokensForTokens(amountIn,amountOutMin,path,addr
ess(this),deadline) (escrow.sol#727-733)
        State variables written after the call(s):
```

```
        State variables written after the call(s):
        - jobCounter ++ (escrow.sol#414)
        - jobList.push(JOBID) (escrow.sol#413)
        - mapJobToAmount[JOBID] = _finalUSD (escrow.sol#410)
Reentrancy in MarketplaceEscrowV1.depositBNB(address,uint256,bytes32,bool) (escrow.sol#323-360):
        External calls:
        - _finalUSD = _pancakeSwapBNB(msg.value) (escrow.sol#347)
                - _amount = PANCAKESWAP.swapExactETHForTokens{value: amountIn}(amountOutMin,path,
address(this),deadline) (escrow.sol#746-748)
        State variables written after the call(s):
        - jobCounter ++ (escrow.sol#356)
        - jobList.push(JOBID) (escrow.sol#355)
        - mapJobToAmount[JOBID] = _finalUSD (escrow.sol#352)
Reentrancy in MarketplaceEscrowV1.releaseAsClient(uint256,bool) (escrow.sol#476-521):
        External calls:
        - _finalRelease = _release(JOBID,_recipient,_amount,true) (escrow.sol#509)
                - require(bool,string)(iBEP20(BLOCK).transferFrom(iBEP20(address(this)).owner(),_
discountRecipient,_amounts[1]),BLOCK_TRANSFER_FAILED) (escrow.sol#655-662)
                - require(bool)(iBEP20(USDT).transfer(_recipient,_finalRelease)) (escrow.sol#674)

        State variables written after the call(s):
        - adminRevenue += (_amount - _finalRelease) (escrow.sol#510)
Reentrancy in MarketplaceEscrowV1.releaseByAdmin(uint256,uint256,uint256) (escrow.sol#556-591):
        External calls:
        - _clientRelease = _release(JOBID,_client,clientSplit,true) (escrow.sol#581)
                - require(bool,string)(iBEP20(BLOCK).transferFrom(iBEP20(address(this)).owner(),_
discountRecipient,_amounts[1]),BLOCK_TRANSFER_FAILED) (escrow.sol#655-662)
                - require(bool)(iBEP20(USDT).transfer(_recipient,_finalRelease)) (escrow.sol#674)
```

```
        - _provRelease = _release(JOBID,_provider,providerSplit,true) (escrow.sol#585)
                - require(bool,string)(iBEP20(BLOCK).transferFrom(iBEP20(address(this)).owner(),_
discountRecipient,_amounts[1]),BLOCK_TRANSFER_FAILED) (escrow.sol#655-662)
                - require(bool)(iBEP20(USDT).transfer(_recipient,_finalRelease)) (escrow.sol#674)

        State variables written after the call(s):
        - adminRevenue += (_amount - _clientRelease - _provRelease) (escrow.sol#587)
Reentrancy in MarketplaceEscrowV1.withdrawQueuedPayment(uint256) (escrow.sol#594-623):
        External calls:
        - _finalRelease = _release(JOBID,_recipient,_amount,true) (escrow.sol#619)
                - require(bool,string)(iBEP20(BLOCK).transferFrom(iBEP20(address(this)).owner(),_
discountRecipient,_amounts[1]),BLOCK_TRANSFER_FAILED) (escrow.sol#655-662)
                - require(bool)(iBEP20(USDT).transfer(_recipient,_finalRelease)) (escrow.sol#674)

        State variables written after the call(s):
        - adminRevenue += (_amount - _finalRelease) (escrow.sol#620)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabiliti
es-2
INFO:Detectors:
Reentrancy in MarketplaceEscrowV1._release(uint256,address,uint256,bool) (escrow.sol#626-677):
        External calls:
        - require(bool,string)(iBEP20(BLOCK).transferFrom(iBEP20(address(this)).owner(),_discount
Recipient,_amounts[1]),BLOCK_TRANSFER_FAILED) (escrow.sol#655-662)
        Event emitted after the call(s):
        - DiscountPaid(_JOBID,_discountRecipient,_amounts[1]) (escrow.sol#663)
Reentrancy in MarketplaceEscrowV1.depositBEP20(address,address,uint256,uint256,bytes32,address[],
bool) (escrow.sol#363-418):
        External calls:
```

```
        External calls:
        - require(bool,string)(iBEP20(asset).transferFrom(msg.sender,address(this),value),Marketp
laceEscrowV1: BEP20_TRANSFER_FAILED) (escrow.sol#376-379)
        - require(bool,string)(iBEP20(asset).approve(ROUTER,value),Swap Router Approval Failure)
(escrow.sol#400-403)
        - _finalUSD = _pancakeSwapTokens(value,swapPath) (escrow.sol#404)
                - _amounts = PANCAKESWAP.swapExactTokensForTokens(amountIn,amountOutMin,path,addr
ess(this),deadline) (escrow.sol#727-733)
        Event emitted after the call(s):
        - Deposit(msg.sender,provider,_finalUSD,JOBID) (escrow.sol#417)
Reentrancy in MarketplaceEscrowV1.depositBNB(address,uint256,bytes32,bool) (escrow.sol#323-360):
        External calls:
        - _finalUSD = _pancakeSwapBNB(msg.value) (escrow.sol#347)
                - _amount = PANCAKESWAP.swapExactETHForTokens{value: amountIn}(amountOutMin,path,
address(this),deadline) (escrow.sol#746-748)
        Event emitted after the call(s):
        - Deposit(msg.sender,provider,_finalUSD,JOBID) (escrow.sol#359)
Reentrancy in MarketplaceEscrowV1.withdrawQueuedPayment(uint256) (escrow.sol#594-623):
        External calls:
        - _finalRelease = _release(JOBID,_recipient,_amount,true) (escrow.sol#619)
                - require(bool,string)(iBEP20(BLOCK).transferFrom(iBEP20(address(this)).owner(),_
discountRecipient,_amounts[1]),BLOCK_TRANSFER_FAILED) (escrow.sol#655-662)
                - require(bool)(iBEP20(USDT).transfer(_recipient,_finalRelease)) (escrow.sol#674)

        Event emitted after the call(s):
        - Release(_client,_recipient,_recipient,_finalRelease,JOBID,1) (escrow.sol#622)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabiliti
es-3
```

```
INFO:Detectors:
MarketplaceEscrowV1.releaseAsClient(uint256,bool) (escrow.sol#476-521) uses timestamp for compari
sons
        Dangerous comparisons:
        - require(bool,string)(mapJobToLockExpiration[JOBID] == 0,JOB_ALREADY_UNDER_REVIEW) (escr
ow.sol#484)
MarketplaceEscrowV1.withdrawQueuedPayment(uint256) (escrow.sol#594-623) uses timestamp for compar
isons
        Dangerous comparisons:
        - require(bool,string)(mapJobToLockExpiration[JOBID] != - 1,Payment already withdrawn) (e
scrow.sol#600-603)
        - require(bool,string)(int256(block.timestamp) >= mapJobToLockExpiration[JOBID],Release T
ime not arrived yet) (escrow.sol#606-609)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
MarketplaceEscrowV1.withdrawQueuedPayment(uint256) (escrow.sol#594-623) compares to a boolean con
stant:
        -require(bool,string)(jobReleased[JOBID] != true,JOB_ALREADY_RELEASED) (escrow.sol#597)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Context._msgData() (escrow.sol#80-83) is never used and should be removed
ReentrancyGuard._reentrancyGuardEntered() (escrow.sol#218-220) is never used and should be remove
d
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.7.5 (escrow.sol#3) allows old versions
solc-0.7.5 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-so
lidity
```

```
INFO:Detectors:
Contract iBEP20 (escrow.sol#49-73) is not in CapWords
Parameter MarketplaceEscrowV1.depositBNB(address,uint256,bytes32,bool).JOBID (escrow.sol#325) is
not in mixedCase
Parameter MarketplaceEscrowV1.depositBEP20(address,address,uint256,uint256,bytes32,address[],bool
).JOBID (escrow.sol#367) is not in mixedCase
Parameter MarketplaceEscrowV1.releaseAsClient(uint256,bool).JOBID (escrow.sol#477) is not in mixe
dCase
Parameter MarketplaceEscrowV1.releaseByProvider(uint256).JOBID (escrow.sol#525) is not in mixedCa
se
Parameter MarketplaceEscrowV1.releaseByAdmin(uint256,uint256,uint256).JOBID (escrow.sol#557) is n
ot in mixedCase
Parameter MarketplaceEscrowV1.withdrawQueuedPayment(uint256).JOBID (escrow.sol#594) is not in mix
edCase
Parameter MarketplaceEscrowV1.raiseDispute(uint256).JOBID (escrow.sol#699) is not in mixedCase
Parameter MarketplaceEscrowV1.viewDiscountUSDT(uint256).JOBID (escrow.sol#825) is not in mixedCas
e
Parameter MarketplaceEscrowV1.isMilestoneEnabled(uint256).JOBID (escrow.sol#833) is not in mixedC
ase
Variable MarketplaceEscrowV1.WBNB (escrow.sol#236) is not in mixedCase
Variable MarketplaceEscrowV1.USDT (escrow.sol#237) is not in mixedCase
Variable MarketplaceEscrowV1.ROUTER (escrow.sol#238) is not in mixedCase
Variable MarketplaceEscrowV1.PANCAKESWAP (escrow.sol#239) is not in mixedCase
Variable MarketplaceEscrowV1.BLOCK (escrow.sol#240) is not in mixedCase
Variable MarketplaceEscrowV1.DEFAULT_FEE_PERCENT (escrow.sol#246) is not in mixedCase
Variable MarketplaceEscrowV1.DEFAULT_DISCOUNT_PERCENT (escrow.sol#247) is not in mixedCase
Variable MarketplaceEscrowV1.TOTAL_MILESTONES (escrow.sol#248) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-
naming-conventions
```

```
INFO:Detectors:
Redundant expression "this (escrow.sol#81)" inContext (escrow.sol#75-84)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,ui
nt256).amountADesired (escrow.sol#36) is too similar to IPancakeRouter01.addLiquidity(address,add
ress,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (escrow.sol#37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-simil
ar
INFO:Detectors:
MarketplaceEscrowV1.TOTAL_MILESTONES (escrow.sol#248) is never used in MarketplaceEscrowV1 (escro
w.sol#223-868)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
MarketplaceEscrowV1.TOTAL_MILESTONES (escrow.sol#248) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-cou
ld-be-declared-constant
INFO:Detectors:
MarketplaceEscrowV1.BLOCK (escrow.sol#240) should be immutable
MarketplaceEscrowV1.USDT (escrow.sol#237) should be immutable
MarketplaceEscrowV1.WBNB (escrow.sol#236) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-cou
ld-be-declared-immutable
INFO:Slither:escrow.sol analyzed (6 contracts with 93 detectors), 47 result(s) found
```

Solidity Static Analysis

BlockToken.sol

-No errors

Escrow.sol

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in MarketplaceEscrowV1._release(uint256,address,uint256,bool): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 416:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in MarketplaceEscrowV1.withdrawRevenue(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 545:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 531:24:

Constant/View/Pure functions:

MarketplaceEscrowV1.getJobs(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 630:4:

Constant/View/Pure functions:

IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint25 : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 28:4:

Similar variable names:

MarketplaceEscrowV1._release(uint256,address,uint256,bool) : Variables have very similar names "_amount" and "_amounts". Note: Modifiers are currently not considered by this static analysis.
Pos: 455:39:

No return:

IPancakeRouter01.getAmountOut(uint256,uint256,uint256): Defines a return type but never explicitly returns a value.
Pos: 39:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 119:8:

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to lost tokens etc.
High	High level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g. public access to crucial functions.
Medium	Medium level vulnerabilities are important to fix; however, they cannot lead to lost tokens.
Low	Low level vulnerabilities are most related to outdated, unused etc. These code snippets cannot have a significant impact on execution.
Lowest Code Style/ Best Practice	Lowest level vulnerabilities, code style violations and information statements cannot affect smart contract execution and can be ignored.

Audit Findings

Critical:

No critical severity vulnerabilities were found.

High:

No high severity vulnerabilities were found.

Medium:

No medium severity vulnerabilities were found.

Low:

No low severity vulnerabilities were found.

Very Low:

No very low severity vulnerabilities were found.

Discussion

Conclusion

We were given a contract file and have used all possible tests based on the given object. so it is ready for mainnet deployment. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

The security state of the reviewed contract is “**Well-Secured**”.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design

documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyse the feasibility of an attack in a live system.

Suggested Solutions

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinised by the developers and deployment engineers, and successful

mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

RD Auditors Disclaimer

The smart contracts given for audit have been analysed in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Because the total number of test cases are unlimited, the audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.

Email: info@rdauditors.com

Website: www.rdauditors.com