SOLIDITY AUDIT 

Smart Contract Solidity Audit

Audit Details:

Audited project: CBOMBER

Deployer Address: 0xcf74ae52ae2c848387e6cd0048e1ec5a93ee2c66

Blockchain: Polygon Matic

April 05

Audit

This document may contain confidential information about IT systems and the 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 a decision of the Customer.

Introduction

Solidity Audit (Consultant) was contracted by Crypto Bomber (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer`s smart contract . Scope The scope of the project is main net smart contracts that can be found on Polyscan:

https://polygonscan.com/token/0xcf74ae52ae2c848387e6cd0048e1ec5a93ee2c66

We have scanned this smart contract for commonly known and more specific vulnerabilities. List of the commonly known vulnerabilities that are considered:

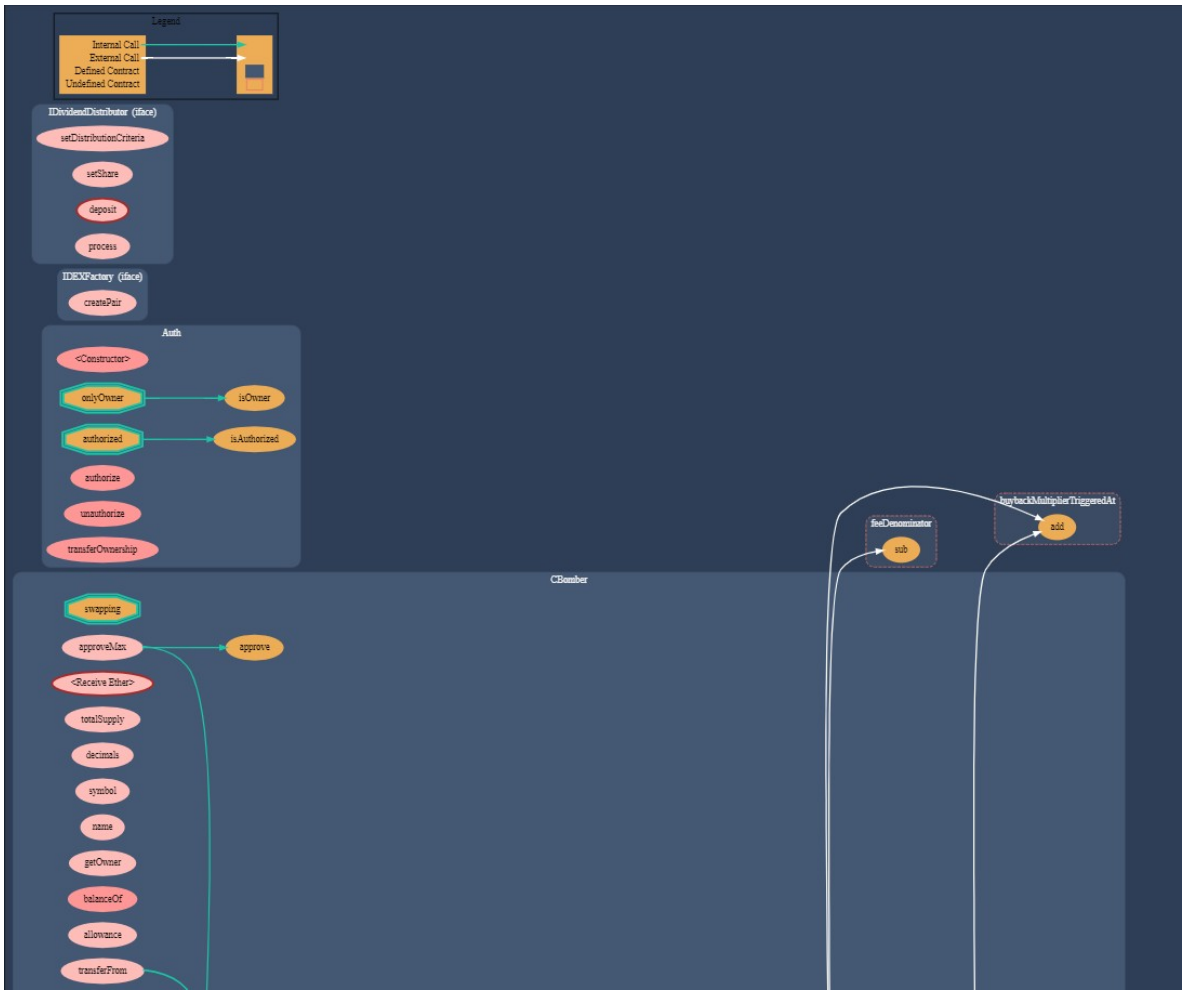
Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency ▪ Data Consistency
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

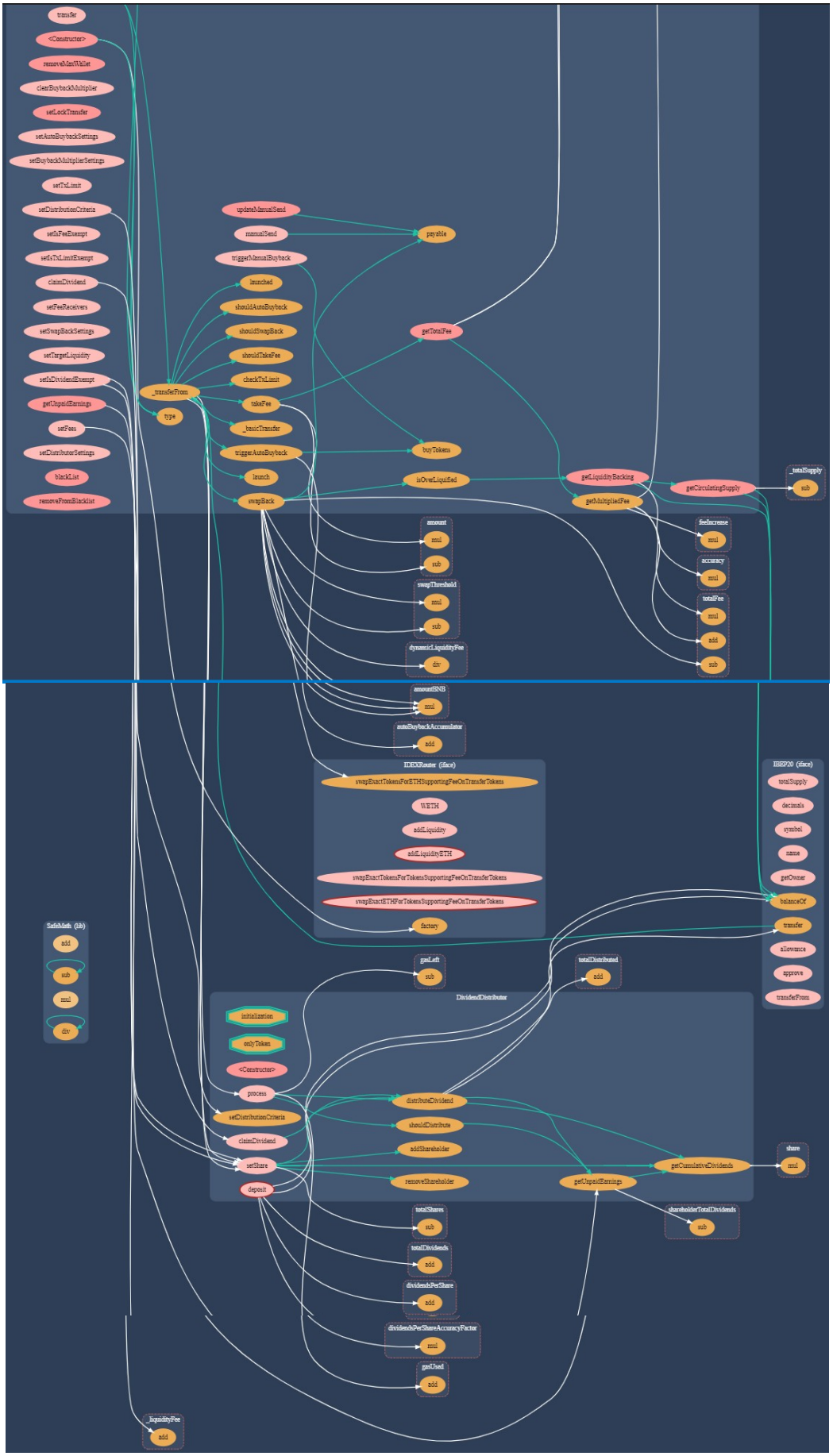
Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. A general overview is presented in AS-IS section, and all found issues can be found in the Audit overview section.

Security engineers found **2** medium, **1** informational issue during the audit.

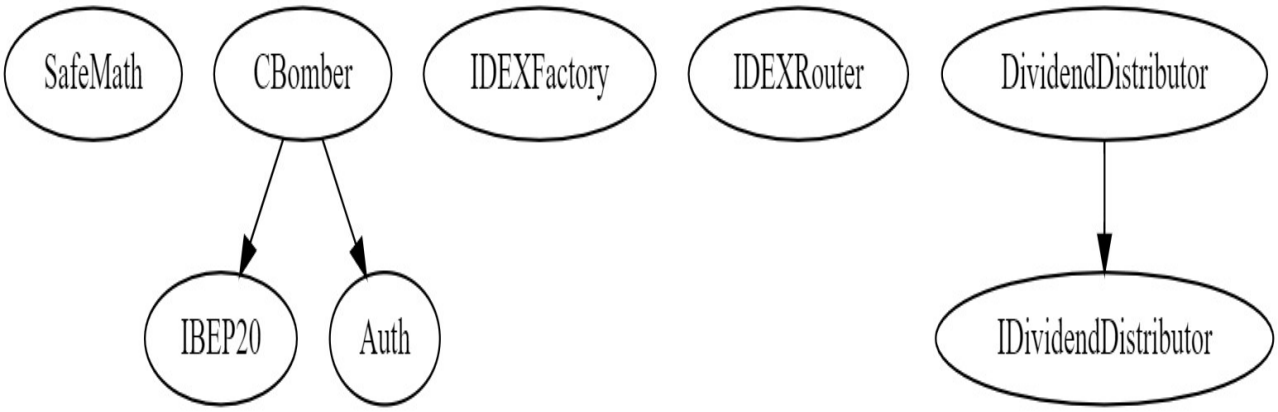
Notice: the audit scope is limited and not include all files in the repository. Though, reviewed contracts are secure, we may not guarantee secureness of contracts that are not in the scope.

ANALYSIS, GRAPHS AND UML DIAGRAM




transfer
<Constructor>
removeAllotWallet
clearBuybackMultiplier
setLockTransfer
setAutoBuybackSettings
setBuybackMultiplierSettings
setTxLimit
setDistributionCriteria
setIsFeeExempt
setIsTxLimitExempt
claimDividend
setFeeReceivers
setSwapBackSettings
setTargetLiquidity
setIsDividendExempt
getUnpaidEarnings
setFees
setDistributorSettings
blackList
removeFromBlacklist
updateManualSend
manualSend
triggerManualBuyback
launched
shouldAutoBuyback
shouldSwapBack
shouldTakeFee
checkTxLimit
takeFee
_basicTransfer
triggerAutoBuyback
launch
swapBack
_transferFrom
type
payable
getTotalFee
buyTokens
isOverLiquified
getLiquidityBacking
getMultipliedFee
getCirculatingSupply
_totalSupply
sub
amount
mul
sub
swapThreshold
mul
sub
dynamicLiquidityFee
div
feeIncrease
mul
accuracy
mul
totalFee
mul
add
sub
amountBNB
mul
autoBuybackAccumulator
add
IDEXRouter (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
addLiquidity
addLiquidityETH
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactETHForTokensSupportingFeeOnTransferTokens
factory
IBEP20 (iface)
totalSupply
decimals
symbol
name
getOwner
balanceOf
transfer
allowance
approve
transferFrom
SafeMath (lib)
add
sub
mul
div
gasLeft
sub
totalDistributed
add
DividendDistributor
initialization
onlyToken
<Constructor>
process
setDistributionCriteria
claimDividend
setShare
deposit
distributeDividend
shouldDistribute
addShareholder
removeShareholder
getCumulativeDividends
getUnpaidEarnings
share
mul
shareholderTotalDividends
sub
totalShares
sub
totalDividends
add
dividendsPerShare
add
dividendsPerShareAccuracyFactor
mul
gasUsed
add
_liquidityFee
add

INHERITANCE



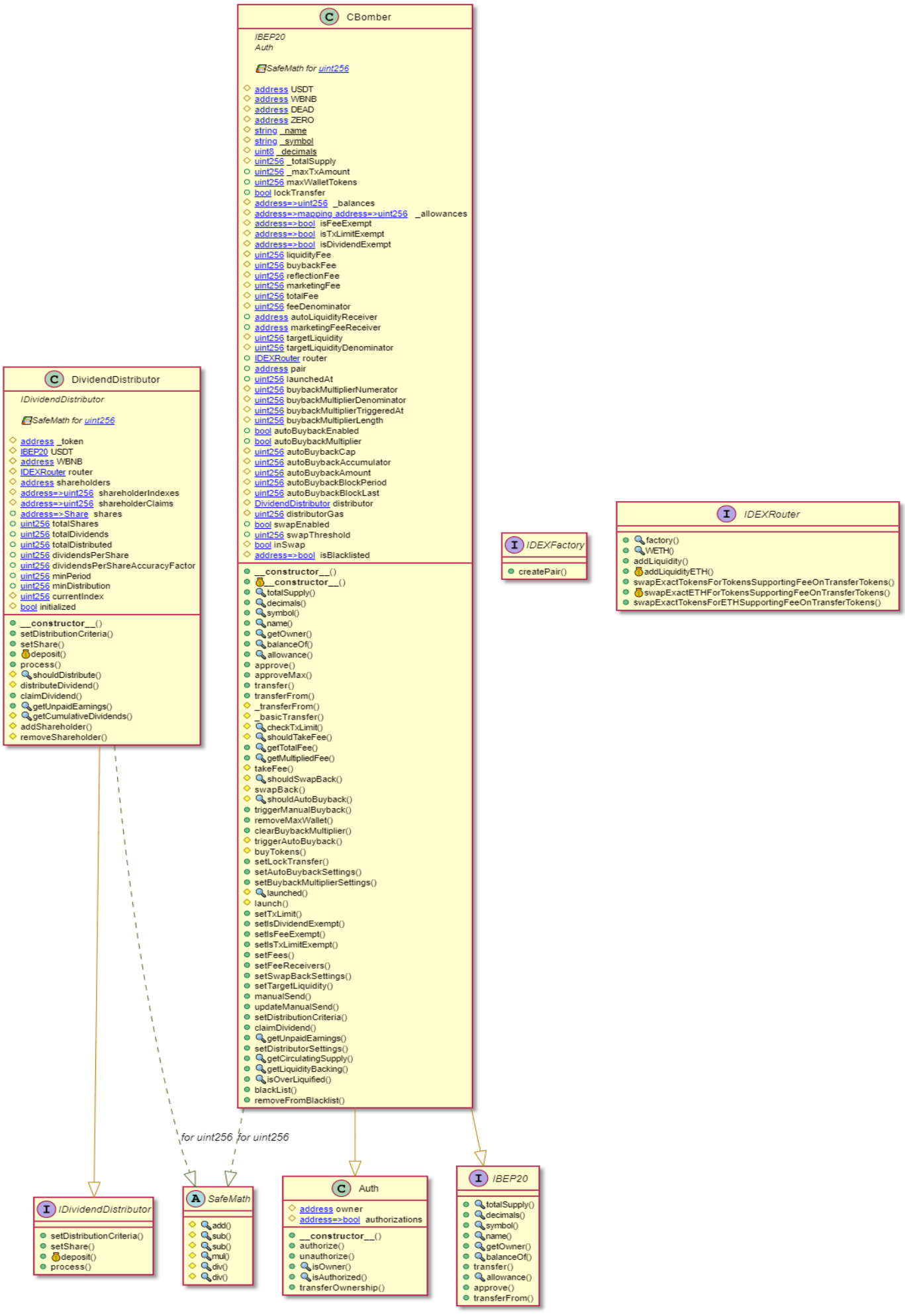
FUNCTION SIGS

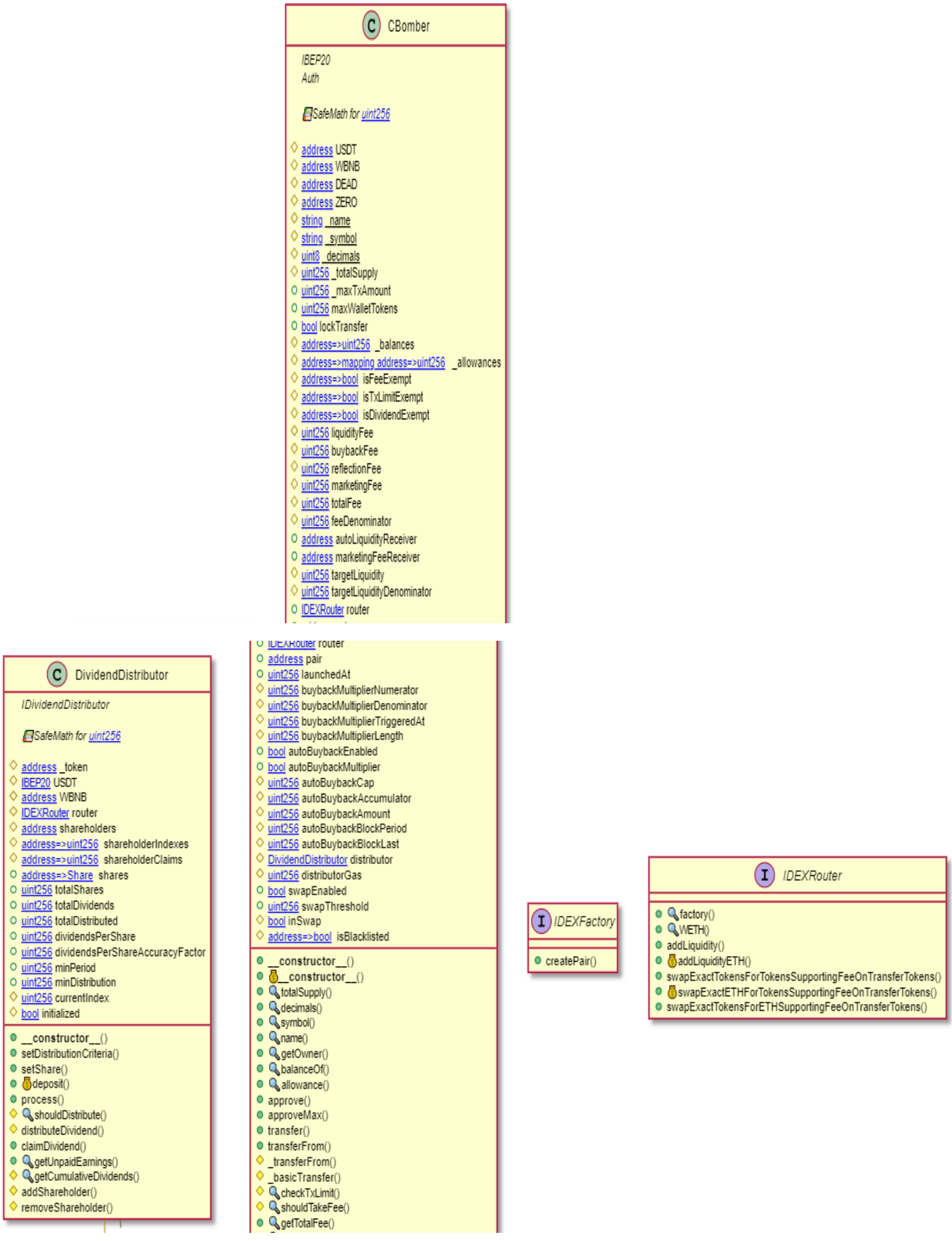
```
Sighash   |   Function Signature
========================
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
e31bdc0a  =>  sub(uint256,uint256,string)
c8a4ac9c  =>  mul(uint256,uint256)
a391c15b  =>  div(uint256,uint256)
b745d336  =>  div(uint256,uint256,string)
18160ddd  =>  totalSupply()
313ce567  =>  decimals()
95d89b41  =>  symbol()
06fdde03  =>  name()
893d20e8  =>  getOwner()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
b6a5d7de  =>  authorize(address)
f0b37c04  =>  unauthorize(address)
2f54bf6e  =>  isOwner(address)
fe9fbb80  =>  isAuthorized(address)
f2fde38b  =>  transferOwnership(address)
c9c65396  =>  createPair(address,address)
```

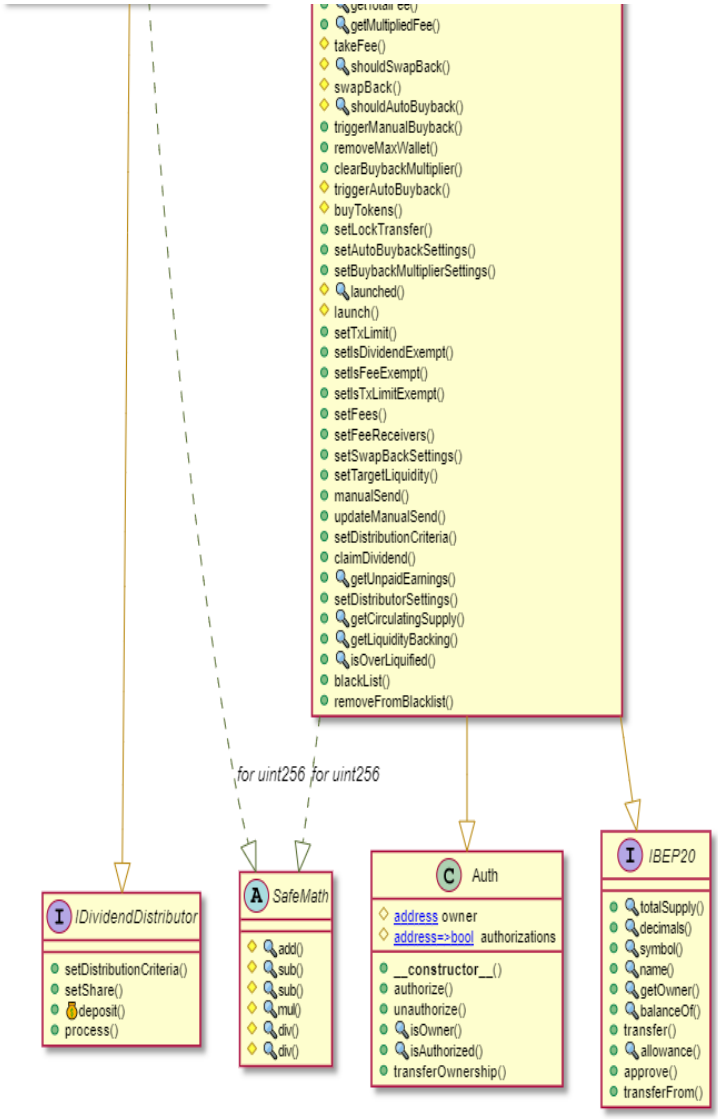
```
c45a0155  =>  factory()
ad5c4648  =>  WETH()
e8e33700  =>
addLiquidity(address,address,uint256,uint256,uint256,uint256,ad-
dress,uint256)
f305d719  =>  addLiquidityETH(address,uint256,uint256,uint256,ad-
dress,uint256)
5c11d795  =>
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint25
6,address[],address,uint256)
b6f9de95  =>
swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[]
,address,uint256)
791ac947  =>
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,a
ddress[],address,uint256)
2d48e896  =>  setDistributionCriteria(uint256,uint256)
14b6ca96  =>  setShare(address,uint256)
d0e30db0  =>  deposit()
ffb2c479  =>  process(uint256)
8c21cd52  =>  shouldDistribute(address)
5319504a  =>  distributeDividend(address)
15f7e05e  =>  claimDividend(address)
28fd3198  =>  getUnpaidEarnings(address)
e68af3ac  =>  getCumulativeDividends(uint256)
db29fe12  =>  addShareholder(address)
9babdad6  =>  removeShareholder(address)
571ac8b0  =>  approveMax(address)
cb712535  =>  _transferFrom(address,address,uint256)
f0774e71  =>  _basicTransfer(address,address,uint256)
4afa518a  =>  checkTxLimit(address,uint256)
e7c44c69  =>  shouldTakeFee(address)
f1f3bca3  =>  getTotalFee(bool)
d806d12f  =>  getMultipliedFee()
20cb7bce  =>  takeFee(address,address,uint256)
0d5c6cea  =>  shouldSwapBack()
6ac5eeee  =>  swapBack()
4d4e6fe5  =>  shouldAutoBuyback()
82334b94  =>  triggerManualBuyback(uint256,bool)
dc07b617  =>  removeMaxWallet()
b210b06d  =>  clearBuybackMultiplier()
5cd44665  =>  triggerAutoBuyback()
c625e9b1  =>  buyTokens(uint256,address)
```

```
c6729f26  =>  setLockTransfer(bool)
2f5620d1  =>
setAutoBuybackSettings(bool,uint256,uint256,uint256,bool)
2375ce40  =>  setBuybackMultiplierSettings(uint256,uint256,uint256)
8091f3bf  =>  launched()
01339c21  =>  launch()
5c85974f  =>  setTxLimit(uint256)
f708a64f  =>  setIsDividendExempt(address,bool)
658d4b7f  =>  setIsFeeExempt(address,bool)
f84ba65d  =>  setIsTxLimitExempt(address,bool)
04a66b48  =>  setFees(uint256,uint256,uint256,uint256,uint256)
a4b45c00  =>  setFeeReceivers(address,address)
df20fd49  =>  setSwapBackSettings(bool,uint256)
201e7991  =>  setTargetLiquidity(uint256,uint256)
f4293890  =>  manualSend()
84134c1f  =>  updateManualSend()
f0fc6bca  =>  claimDividend()
9d1944f5  =>  setDistributorSettings(uint256)
2b112e49  =>  getCirculatingSupply()
d51ed1c8  =>  getLiquidityBacking(uint256)
1161ae39  =>  isOverLiquified(uint256,uint256)
4838d165  =>  blackList(address)
537df3b6  =>  removeFromBlacklist(address)
```

UML


CBomber
IBEP20
Auth
SafeMath for uint256
address USDT
address WBNB
address DEAD
address ZERO
string _name
string _symbol
uint8 _decimals
uint256 _totalSupply
uint256 _maxTxAmount
uint256 maxWalletTokens
bool lockTransfer
address=>uint256 _balances
address=>mapping address=>uint256 _allowances
address=>bool isFeeExempt
address=>bool isTxLimitExempt
address=>bool isDividendExempt
uint256 liquidityFee
uint256 buybackFee
uint256 reflectionFee
uint256 marketingFee
uint256 totalFee
uint256 feeDenominator
address autoLiquidityReceiver
address marketingFeeReceiver
uint256 targetLiquidity
uint256 targetLiquidityDenominator
IDEXRouter router
address pair
uint256 launchedAt
uint256 buybackMultiplierNumerator
uint256 buybackMultiplierDenominator
uint256 buybackMultiplierTriggeredAt
uint256 buybackMultiplierLength
bool autoBuybackEnabled
bool autoBuybackMultiplier
uint256 autoBuybackCap
uint256 autoBuybackAccumulator
uint256 autoBuybackAmount
uint256 autoBuybackBlockPeriod
uint256 autoBuybackBlockLast
DividendDistributor distributor
uint256 distributorGas
bool swapEnabled
uint256 swapThreshold
bool inSwap
address=>bool isBlacklisted
__constructor__()
__constructor__()
totalSupply()
decimals()
symbol()
name()
getOwner()
balanceOf()
allowance()
approve()
approveMax()
transfer()
transferFrom()
_transferFrom()
_basicTransfer()
checkTxLimit()
shouldTakeFee()
getTotalFee()
getMultipliedFee()
takeFee()
shouldSwapBack()
swapBack()
shouldAutoBuyback()
triggerManualBuyback()
removeMaxWallet()
clearBuybackMultiplier()
triggerAutoBuyback()
buyTokens()
setLockTransfer()
setAutoBuybackSettings()
setBuybackMultiplierSettings()
launched()
launch()
setTxLimit()
setIsDividendExempt()
setIsFeeExempt()
setIsTxLimitExempt()
setFees()
setFeeReceivers()
setSwapBackSettings()
setTargetLiquidity()
manualSend()
updateManualSend()
setDistributionCriteria()
claimDividend()
getUnpaidEarnings()
setDistributorSettings()
getCirculatingSupply()
getLiquidityBacking()
isOverLiquified()
blackList()
removeFromBlacklist()
DividendDistributor
IDividendDistributor
SafeMath for uint256
address _token
IBEP20 USDT
address WBNB
IDEXRouter router
address shareholders
address=>uint256 shareholderIndexes
address=>uint256 shareholderClaims
address=>Share shares
uint256 totalShares
uint256 totalDividends
uint256 totalDistributed
uint256 dividendsPerShare
uint256 dividendsPerShareAccuracyFactor
uint256 minPeriod
uint256 minDistribution
uint256 currentIndex
bool initialized
__constructor__()
setDistributionCriteria()
setShare()
deposit()
process()
shouldDistribute()
distributeDividend()
claimDividend()
getUnpaidEarnings()
getCumulativeDividends()
addShareholder()
removeShareholder()
IDEXFactory
createPair()
IDEXRouter
factory()
WETH()
addLiquidity()
addLiquidityETH()
swapExactTokensForTokensSupportingFeeOnTransferTokens()
swapExactETHForTokensSupportingFeeOnTransferTokens()
swapExactTokensForETHSupportingFeeOnTransferTokens()
for uint256
for uint256
IDividendDistributor
setDistributionCriteria()
setShare()
deposit()
process()
SafeMath
add()
sub()
sub()
mul()
div()
div()
Auth
address owner
address=>bool authorizations
__constructor__()
authorize()
unauthorize()
isOwner()
isAuthorized()
transferOwnership()
IBEP20
totalSupply()
decimals()
symbol()
name()
getOwner()
balanceOf()
transfer()
allowance()
approve()
transferFrom()


CBomber
IBEP20
Auth
SafeMath for uint256
address USDT
address WBNB
address DEAD
address ZERO
string _name
string _symbol
uint8 _decimals
uint256 _totalSupply
uint256 _maxTxAmount
uint256 maxWalletTokens
bool lockTransfer
address=>uint256 _balances
address=>mapping address=>uint256 _allowances
address=>bool isFeeExempt
address=>bool isTxLimitExempt
address=>bool isDividendExempt
uint256 liquidityFee
uint256 buybackFee
uint256 reflectionFee
uint256 marketingFee
uint256 totalFee
uint256 feeDenominator
address autoLiquidityReceiver
address marketingFeeReceiver
uint256 targetLiquidity
uint256 targetLiquidityDenominator
IDEXRouter router
IDEXRouter router
address pair
uint256 launchedAt
uint256 buybackMultiplierNumerator
uint256 buybackMultiplierDenominator
uint256 buybackMultiplierTriggeredAt
uint256 buybackMultiplierLength
bool autoBuybackEnabled
bool autoBuybackMultiplier
uint256 autoBuybackCap
uint256 autoBuybackAccumulator
uint256 autoBuybackAmount
uint256 autoBuybackBlockPeriod
uint256 autoBuybackBlockLast
DividendDistributor distributor
uint256 distributorGas
bool swapEnabled
uint256 swapThreshold
bool inSwap
address=>bool isBlacklisted
__constructor__()
__constructor__()
totalSupply()
decimals()
symbol()
name()
getOwner()
balanceOf()
allowance()
approve()
approveMax()
transfer()
transferFrom()
_transferFrom()
_basicTransfer()
checkTxLimit()
shouldTakeFee()
getTotalFee()
DividendDistributor
IDividendDistributor
SafeMath for uint256
address _token
IBEP20 USDT
address WBNB
IDEXRouter router
address shareholders
address=>uint256 shareholderIndexes
address=>uint256 shareholderClaims
address=>Share shares
uint256 totalShares
uint256 totalDividends
uint256 totalDistributed
uint256 dividendsPerShare
uint256 dividendsPerShareAccuracyFactor
uint256 minPeriod
uint256 minDistribution
uint256 currentIndex
bool initialized
__constructor__()
setDistributionCriteria()
setShare()
deposit()
process()
shouldDistribute()
distributeDividend()
claimDividend()
getUnpaidEarnings()
getCumulativeDividends()
addShareholder()
removeShareholder()
IDEXFactory
createPair()
IDEXRouter
factory()
WETH()
addLiquidity()
addLiquidityETH()
swapExactTokensForTokensSupportingFeeOnTransferTokens()
swapExactETHForTokensSupportingFeeOnTransferTokens()
swapExactTokensForETHSupportingFeeOnTransferTokens()


getMultipliedFee()
takeFee()
shouldSwapBack()
swapBack()
shouldAutoBuyback()
triggerManualBuyback()
removeMaxWallet()
clearBuybackMultiplier()
triggerAutoBuyback()
buyTokens()
setLockTransfer()
setAutoBuybackSettings()
setBuybackMultiplierSettings()
launched()
launch()
setTxLimit()
setIsDividendExempt()
setIsFeeExempt()
setIsTxLimitExempt()
setFees()
setFeeReceivers()
setSwapBackSettings()
setTargetLiquidity()
manualSend()
updateManualSend()
setDistributionCriteria()
claimDividend()
getUnpaidEarnings()
setDistributorSettings()
getCirculatingSupply()
getLiquidityBacking()
isOverLiquified()
blackList()
removeFromBlacklist()
for uint256
for uint256
IDividendDistributor
setDistributionCriteria()
setShare()
deposit()
process()
SafeMath
add()
sub()
sub()
mul()
div()
div()
Auth
address owner
address=>bool authorizations
__constructor__()
authorize()
unauthorize()
isOwner()
isAuthorized()
transferOwnership()
IBEP20
totalSupply()
decimals()
symbol()
name()
getOwner()
balanceOf()
transfer()
allowance()
approve()
transferFrom()

Solidity Audit Disclaimer

The smart contracts given for audit have been analyzed 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). The audit makes no statements or warranties on 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 blockchain platform. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.