CONTRACT WOLF

Blockchain Security - Smart Contract Audits

Security Assessment

May 22, 2022

Disclaimer

ContractWolf.io audits and reports should not be considered as a form of project's "advertisement" and does not cover any interaction and assessment from "project's contract" to "external contracts" such as Pancakeswap or similar.

ContractWolf does not provide any warranty on its released reports.

ContractWolf should not be used as a decision to invest into an audited project and is not affiliated nor partners to its audited contract projects.

ContractWolf provides transparent report to all its "clients" and to its "clients participants" and will not claim any guarantee of bug-free code within its **SMART CONTRACT**.

ContractWolf presence is to analyze, audit and assess the client's smart contract's code.

Each company or projects should be liable to its security flaws and functionalities.

Scope of Work

Metal Shiba Inu team agreed and provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract.

The goal of this engagement was to identify if there is a possibility of security flaws in the implementation of the contract or system.

ContractWolf will be focusing on contract issues and functionalities along with the projects claims from smart contract to their website, whitepaper and repository which has been provided by **Metal Shiba Inu.**

Network

Binance Smart Chain (BEP20)

Contract link

https://bscscan.com/address/0x870161f68C79385692C17EBd809537B1f7082165

Website

https://metalshiba.com/

Telegram

https://t.me/MetalShibaInu

Twitter

https://twitter.com/METALSHIBAINU

Instagram

https://www.instagram.com/metalshibainu/

Discord

https://discord.gg/TtsvPHH2Mt

Gitbook

https://metalshiba-inu.gitbook.io/whitepaper/

Description

Metal Shiba Inu is the ultimate superhero inspired by the Shiba meme. The idea is to then create a paper version and promote it at the most famous comic book international fairs (such as Lucca Comics). This will benefit not only the comic itself but also the token and community built around it.

Logo

Risk Level Classification

Risk Level represents the classification or the probability that a certain function or threat that can exploit vulnerability and have an impact within the system or contract.
Risk Level is computed based on CVSS Version 3.0

Level	Value	Vulnerability
Critical	9 - 10	An Exposure that can affect the contract functions in several events that can risk and disrupt the contract
High	7 - 8.9	An Exposure that can affect the outcome when using the contract that can serve as an opening in manipulating the contract in an unwanted manner
Medium	4 - 6.9	An opening that could affect the outcome in executing the contract in a specific situation
Low	0.1 - 3.9	An opening but doesn't have an impact on the functionality of the contract
Informational	0	An opening that consists of information's but will not risk or affect the contract

Auditing Approach

Every line of code along with its functionalities will undergo manual review to check its security issues, quality, and contract scope of inheritance. The manual review will be done by our team that will document any issues that there were discovered.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:

    - Review of the specifications, sources, and instructions provided to ContractWolf to make sure we understand the size, scope, and functionality of the smart contract.
    - Manual review of code, our team will have a process of reading the code line-by-line with the intention of identifying potential vulnerabilities and security flaws.

2. Testing and automated analysis that includes:

    - Testing the smart contract functions with common test cases and scenarios, to ensure that it returns the expected results.

3. Best practices review, the team will review the contract with the aim to improve efficiency, effectiveness, clarifications, maintainability, security, and control within the smart contract.

4. Recommendations to help the project take steps to secure the smart contract.

Used Code from other Frameworks/Smart Contracts (Direct Imports)

Imported Packages

- IBEP20
- Ownable
- TimeLock
- PCSFactory
- PCSv2Router
- PCSv2Pair
- MetalShiba

Description

Optimization enabled: Yes

Decimal: 18

Symbol: METAL

Max / Total supply: 21,000,000,000

Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	1	0	4	2

Exposed Functions

Version	Public	Private	External	Internal
1.0	16	0	74	9

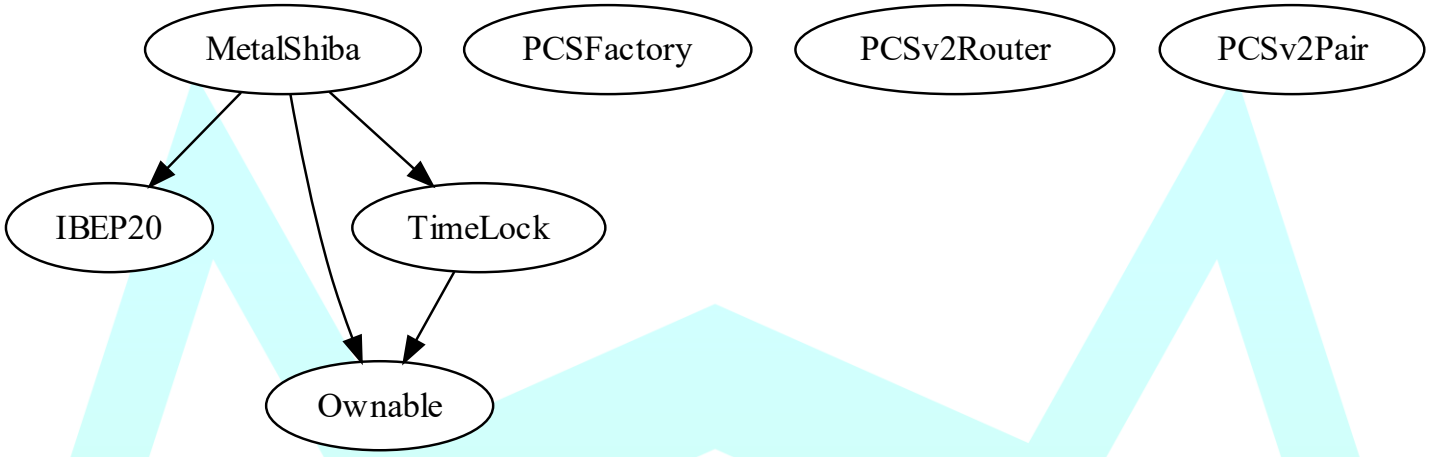
State Variables

Version	Total	Public
1.0	84	55

Capabilities

Version	Solidity Versions Observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	v0.8.14		Yes	Yes	No

Inheritance Graph



Correct implementation of Token Standard

Tested	Verified
✓	✗

Overall Checkup (Smart Contract Security)

Tested	Verified
✓	✓

Function	Description	Exist	Tested	Verified
TotalSupply	Information about the total coin or token supply	✓	✓	✓
BalanceOf	Details on the account balance from a specified address	✓	✓	✓
Transfer	An action that transfers a specified amount of coin or token to a specified address	✓	✓	✓
TransferFrom	An action that transfers a specified amount of coin or token from a specified address	✓	✓	✓
Approve	Provides permission to withdraw specified number of coin or token from a specified address	✓	✓	✓

Verify Claims

Statement	Exist	Tested	Deployer
Renounce Ownership	–	–	–
Mint	–	–	–
Burn	–	–	–
Block	–	–	–
Pause	–	–	–

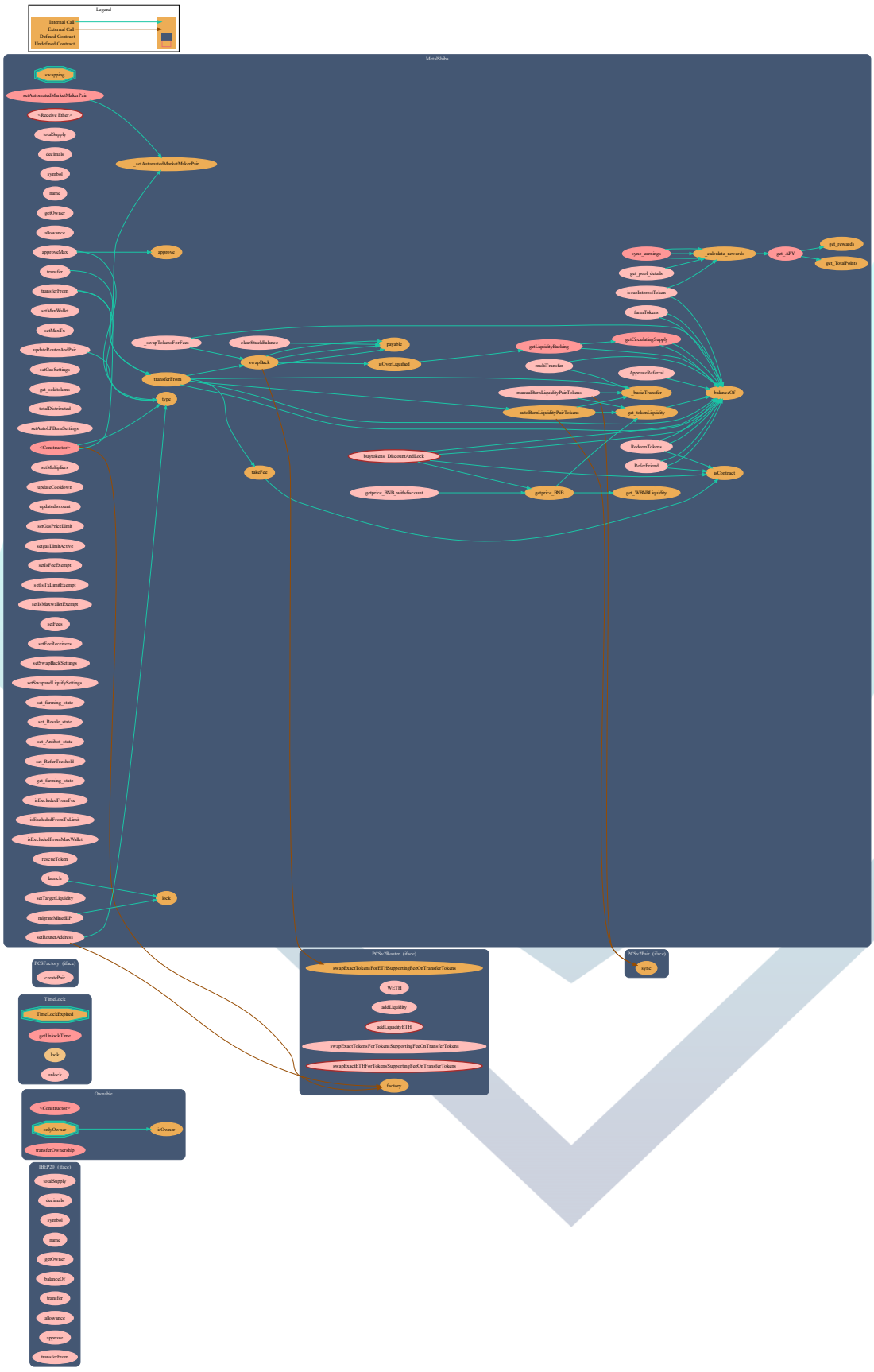
Legend

Attribute	Symbol
Verified / Can	✓
Verified / Cannot	✗
Unverified / Not checked	⚑
Not Available	–

Write Functions of Contract

1. ApproveReferral
2. RedeemTokens
3. ReferFriend
4. _swapTokensForFees
5. approve
6. approveMax
7. buytokens_DiscountAndLock
8. clearStuckBalance
9. farmTokens
10. issueInterestToken
11. launch
12. manualBurnLiquidityPairTokens
13. migrateMinedLP
14. multiTransfer
15. rescueToken
16. setAutoLPBurnSettings
17. setAutomatedMarketMakerPair
18. setFeeReceivers
19. setFees
20. setGasPriceLimit
21. setGasSettings
22. setIsFeeExempt
23. setIsMaxwalletExempt
24. setIsTxLimitExempt
25. setMaxTx
26. setMaxWallet
27. setMultipliers
28. setRouterAddress
29. setSwapBackSettings
30. setSwapandLiquifySettings
31. setTargetLiquidity
32. set_Antibot_state
33. set_ReferTreshold
34. set_Resale_state
35. set_farming_state
36. setgasLimitActive
37. sync_earnings
38. transfer
39. transferFrom
40. transferOwnership
41. unlock
42. updateCooldown
43. updateRouterAndPair
44. updatediscount

Call Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
MetaShiba
swapping
setAutomatedMarketMakerPair
<Receive Ether>
totalSupply
decimals
symbol
name
getOwner
allowance
approveMax
transfer
transferFrom
setMaxWallet
setMaxTx
updateRouterAndPair
setGasSettings
get_soldtokens
totalDistributed
setAutoLPBurnSettings
<Constructor>
setMultipliers
updateCooldown
updatefeecount
setGasPriceLimit
setgasLimitActive
setIsFeeExempt
setIsTxLimitExempt
setIsMaxwalletExempt
setFees
setFeeReceivers
setSwapbackSettings
setSwapandLiquifySettings
set_farming_state
set_Resale_state
set_Antibot_state
set_ReferTreshold
get_farming_state
isExcludedFromFee
isExcludedFromTxLimit
isExcludedFromMaxWallet
rescueToken
launch
setTargetLiquidity
migrateMetaLP
setRouterAddress
_setAutomatedMarketMakerPair
approve
_swapTokensForFees
clearStuckBalance
swapBack
payable
isOverLiquified
_transferFrom
type
takeFee
buytokens_DiscountAndLock
getprice_BNB_withdiscount
getprice_BNB
sync_earnings
_calculate_rewards
get_APY
get_rewards
get_TotalPoints
get_pool_details
investInterestToken
farmTokens
getCirculatingSupply
getLiquidityBacking
multiTransfer
ApproveReferral
manualBurnLiquidityPairTokens
autoBurnLiquidityPairTokens
basicTransfer
get_tokenLiquidity
balanceOf
RedeemTokens
ReferFriend
isContract
get_WBNBLiquidity
PCSv2Router (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
addLiquidity
addLiquidityETH
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactETHForTokensSupportingFeeOnTransferTokens
factory
PCSv2Pair (iface)
sync
PCSFactory (iface)
createPair
TimeLock
TimeLockExpired
getUnlockTime
lock
unlock
Ownable
<Constructor>
onlyOwner
isOwner
transferOwnership
IBEP20 (iface)
totalSupply
decimals
symbol
name
getOwner
balanceOf
transfer
allowance
approve
transferFrom

SWC Attacks

ID	Title	Status
SWC-136	Unencrypted Private Data On-Chain	PASSED
SWC-135	Code With No Effects	PASSED
SWC-134	Message call with hardcoded gas amount	PASSED
SWC-133	Hash Collisions with Multiple Variable Length Arguments	PASSED
SWC-132	Unexpected Ether balance	PASSED
SWC-131	Presence of unused variables	PASSED
SWC-130	Right-To Left Override control character (U+202E)	PASSED
SWC-129	Typographical Error	PASSED
SWC-128	DoS With Block Gas Limit	PASSED
SWC-127	Arbitrary Jump with Function Type Variable	PASSED
SWC-126	Insufficient Gas Griefing	PASSED
SWC-125	Incorrect Inheritance Order	PASSED
SWC-124	Write to Arbitrary Storage Location	PASSED
SWC-123	Requirement Violation	PASSED
SWC-122	Lack of Proper Signature Verification	PASSED
SWC-121	Missing Protection against Signature Replay Attacks	PASSED
SWC-120	Weak Sources of Randomness from Chain Attributes	LOW ISSUE
SWC-119	Shadowing State Variables	PASSED
SWC-118	Incorrect Constructor Name	PASSED
SWC-117	Signature Malleability	PASSED
SWC-116	Block values as a proxy for time	PASSED
SWC-115	Authorization through tx.origin	PASSED
SWC-114	Transaction Order Dependence	PASSED
SWC-113	DoS with Failed Call	PASSED

SWC-112	Delegate call to Untrusted Callee	**PASSED**
SWC-111	Use of Deprecated Solidity Functions	**PASSED**
SWC-110	Assert Violation	**PASSED**
SWC-109	Uninitialized Storage Pointer	**PASSED**
SWC-108	State Variable Default Visibility	**PASSED**
SWC-107	Reentrancy	**PASSED**
SWC-106	Unprotected SELFDESTRUCT Instruction	**PASSED**
SWC-105	Unprotected Ether Withdrawal	**PASSED**
SWC-104	Unchecked Call Return Value	**PASSED**
SWC-103	Floating Pragma	**PASSED**
SWC-102	Outdated Compiler Version	**PASSED**
SWC-101	Integer Overflow and Underflow	**PASSED**
SWC-100	Function Default Visibility	**PASSED**

AUDIT PASSED

Low Issues

Potential use of “block.number” as source of randomness (SWC-120)	L: 366 C: 34, L: 448 C: 20, L: 451 C: 16, L: 605 C: 4

Audit Comments

- Deployer can transfer ownership
- Deployer can set max wallet with an indefinite amount
- Deployer can set max transaction limit with an indefinite amount
- Deployer can set/toggle automated market maker pair
- Deployer can set auto burn LP settings
- Deployer can set sell/buy/transfer multipliers not greater than 20
- Deployer can set gas price with an amount not greater than 750000
- Deployer can set gas price limit with an amount not less than 20
- Deployer can set fees not greater than 10%
- Deployer can set fee receivers
- Deployer can set swap back settings
- Deployer can set swap and liquify settings
- Deployer can set refer threshold
- Deployer can set target liquidity
- Deployer can set router address
- Deployer can update cooldown time
- Deployer can update discount not greater than 50%
- Deployer can update router and pair addresses
- Deployer can toggle gas limit status
- Deployer can toggle farming status
- Deployer can toggle resale status
- Deployer can toggle antibot status
- Deployer can include/exclude addresses from fees
- Deployer can include/exclude addresses from transaction limit
- Deployer can include/exclude addresses from max wallet limit
- Deployer can launch

- Deployer can unlock function after 180 days
- Deployer can airdrop to multiple addresses up to 500 addresses
- Deployer can collect BNB/tokens from contract
- Deployer can swap token for fees
- Deployer can manually burn liquidity pair tokens with an amount not greater than 10%
- Deployer can migrate mined LP tokens and lock contract for 180 days
- Deployer cannot burn
- Deployer cannot block user
- Deployer cannot pause contract
- Deployer cannot mint after initial deployment

CONTRACTWOLF

Blockchain Security - Smart Contract Audits