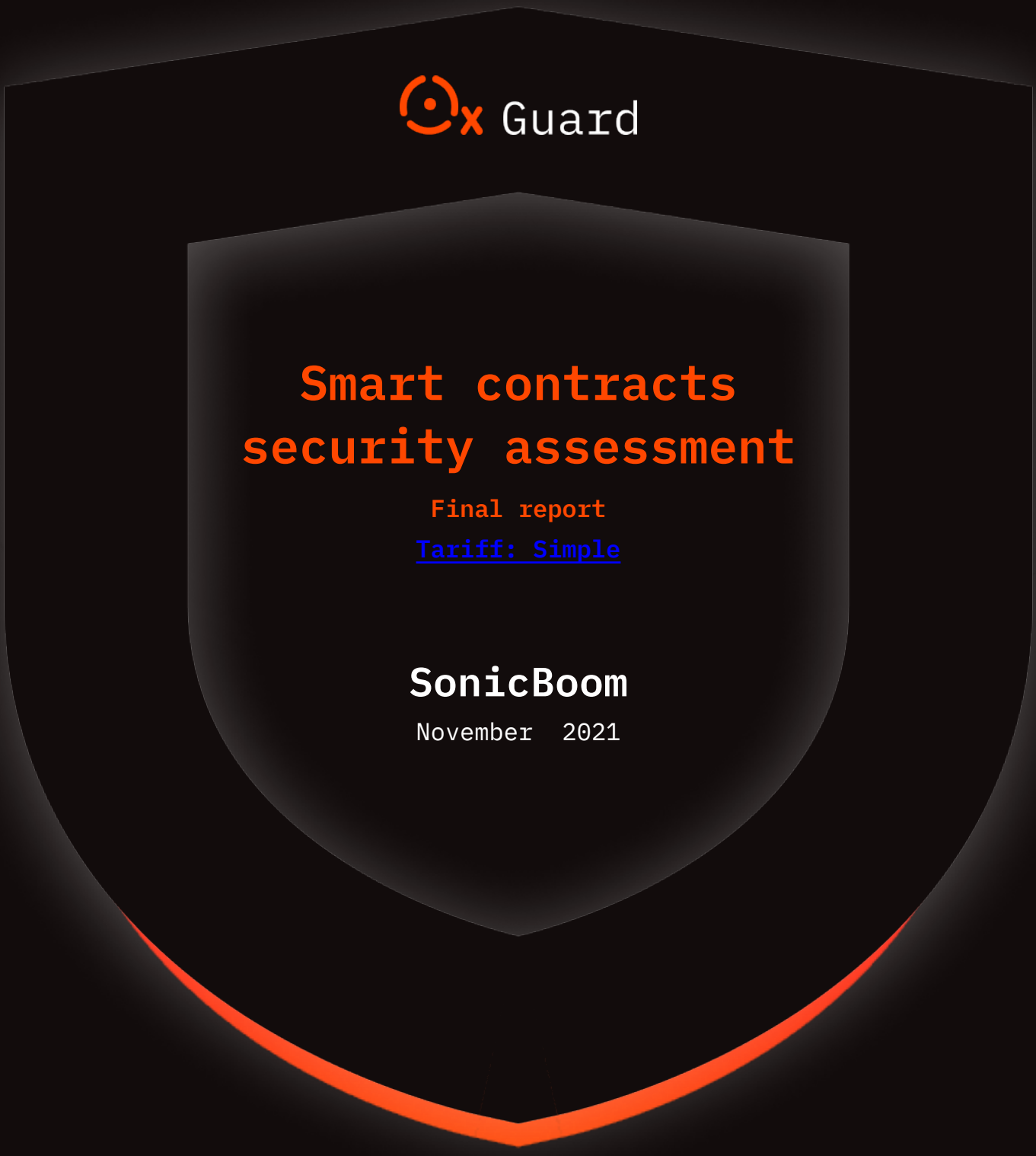

Guard

Smart contracts security assessment

Final report

Tariff: Simple

SonicBoom

November 2021

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for SonicBoom

Name	SonicBoom
Audit date	2021-11-02 - 2021-11-02
Language	Solidity
Platform	Polygon Network

Contracts checked

Name	Address
SonicToken	0x93f1005807e033D865e4e5a167d198e1b64d305b
MasterChef	0xB91e6A88d1fbE3E9225cB962d010b02a75ca32A1

Known vulnerabilities checked

Title	Check result
Unencrypted Private Data On-Chain	passed
Code With No Effects	passed
Message call with hardcoded gas amount	passed
Typographical Error	passed
DoS With Block Gas Limit	passed
Presence of unused variables	Not passed
Typographical Error	passed
Incorrect Inheritance Order	passed
Requirement Violation	passed

Weak Sources of Randomness from Chain Attributes	passed
Shadowing State Variables	passed
Incorrect Constructor Name	passed
Block values as a proxy for time	passed
Authorization through tx.origin	passed
DoS with Failed Call	passed
Delegatecall to Untrusted Callee	passed
Use of Deprecated Solidity Functions	passed
Assert Violation	passed
State Variable Default Visibility	passed
Reentrancy	passed
Unprotected SELFDESTRUCT Instruction	passed
Unprotected Ether Withdrawal	passed
Unchecked Call Return Value	passed
Floating Pragma	passed
Outdated Compiler Version	passed
Integer Overflow and Underflow	passed
Function Default Visibility	passed

Classification of issue severity

High severity — High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.

Medium severity	Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.
Low severity	Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

1. Broken governance mechanism (SonicToken)

Function _moveDelegates (L1316) is not used within the token transfer methods.

Medium severity issues

1. Unrestricted for-loop (SonicToken)

L1544: function massUpdatePools contains a for-loop which is not restricted by amount of iterations.

2. Emission rate not limited (SonicToken)

In function updateEmissionRate the owner can set an arbitrary big value for emission rate. The emission rate must be capped.

Low severity issues

No issues were found

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability)set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without 0xGuard prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts 0xGuard to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Static code analysis result

```
Compilation warnings/errors on ./contract.sol:
Warning: Documentation tag on non-public state variables will be disallowed in 0.7.0.
You will need to use the @dev tag explicitly.
    --> ./contract.sol:1143:5:
     |
1143 |     /// @notice A record of each accounts delegate
     |     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^


MasterChef.safeSonicTransfer(address,uint256) (contract.sol#1652-1659) ignores return
value by sonic.transfer(_to,sonicBal) (contract.sol#1655)
MasterChef.safeSonicTransfer(address,uint256) (contract.sol#1652-1659) ignores return
value by sonic.transfer(_to,_amount) (contract.sol#1657)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-
transfer

MasterChef.pendingSonic(uint256,address) (contract.sol#1523-1535) performs a
multiplication on the result of a division:
-sonicReward = multiplier.mul(sonicPerBlock).mul(pool.allocPoint).div(totalAllocPoint)
(contract.sol#1530)
-accSonicPerShare = accSonicPerShare.add(sonicReward.mul(1e12).div(lpSupply))
(contract.sol#1531)
MasterChef.updatePool(uint256) (contract.sol#1552-1568) performs a multiplication on
the result of a division:
-sonicReward = multiplier.mul(sonicPerBlock).mul(pool.allocPoint).div(totalAllocPoint)
(contract.sol#1563)
-pool.accSonicPerShare =
pool.accSonicPerShare.add(sonicReward.mul(1e12).div(lpSupply)) (contract.sol#1566)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply

SonicToken._writeCheckpoint(address,uint32,uint256,uint256) (contract.sol#1336-1354)
uses a dangerous strict equality:
- nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock ==
blockNumber (contract.sol#1346)
MasterChef.updatePool(uint256) (contract.sol#1552-1568) uses a dangerous strict
equality:
```

```
- lpSupply == 0 || pool.allocPoint == 0 (contract.sol#1558)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-
strict-equalities

Reentrancy in MasterChef.add(uint256,IBEP20,uint16,uint256,bool)
(contract.sol#1485-1502):
External calls:
- massUpdatePools() (contract.sol#1489)
  - sonic.mint(devAddress,sonicReward.div(10)) (contract.sol#1564)
  - sonic.mint(address(this),sonicReward) (contract.sol#1565)
State variables written after the call(s):
- poolExistence[_lpToken] = true (contract.sol#1493)
- poolInfo.push(PoolInfo(_lpToken,_allocPoint,lastRewardBlock,0,_depositFeeBP,_harvestI
nterval)) (contract.sol#1494-1501)
- totalAllocPoint = totalAllocPoint.add(_allocPoint) (contract.sol#1492)
Reentrancy in MasterChef.deposit(uint256,uint256,address) (contract.sol#1571-1591):
External calls:
- updatePool(_pid) (contract.sol#1574)
  - sonic.mint(devAddress,sonicReward.div(10)) (contract.sol#1564)
  - sonic.mint(address(this),sonicReward) (contract.sol#1565)
- sonicBoomReferral.recordReferral(msg.sender,_referrer) (contract.sol#1576)
- payOrLockuppendingSonic(_pid) (contract.sol#1578)
  - sonic.transfer(_to,sonicBal) (contract.sol#1655)
  - sonic.transfer(_to,_amount) (contract.sol#1657)
  - sonic.mint(referrer,commissionAmount) (contract.sol#1699)
  - sonicBoomReferral.recordReferralCommission(referrer,commissionAmount)
(contract.sol#1700)
State variables written after the call(s):
- payOrLockuppendingSonic(_pid) (contract.sol#1578)
  - user.nextHarvestUntil = block.timestamp.add(pool.harvestInterval)
(contract.sol#1627)
  - user.rewardLockedUp = 0 (contract.sol#1637)
  - user.nextHarvestUntil = block.timestamp.add(pool.harvestInterval)
(contract.sol#1638)
  - user.rewardLockedUp = user.rewardLockedUp.add(pending) (contract.sol#1645)
Reentrancy in MasterChef.deposit(uint256,uint256,address) (contract.sol#1571-1591):
External calls:
- updatePool(_pid) (contract.sol#1574)
  - sonic.mint(devAddress,sonicReward.div(10)) (contract.sol#1564)
  - sonic.mint(address(this),sonicReward) (contract.sol#1565)
- sonicBoomReferral.recordReferral(msg.sender,_referrer) (contract.sol#1576)
```

```
- payOrLockuppendingSonic(_pid) (contract.sol#1578)
  - sonic.transfer(_to,sonicBal) (contract.sol#1655)
  - sonic.transfer(_to,_amount) (contract.sol#1657)
  - sonic.mint(referrer,commissionAmount) (contract.sol#1699)
  - sonicBoomReferral.recordReferralCommission(referrer,commissionAmount)
(contract.sol#1700)
- pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
(contract.sol#1580)
- pool.lpToken.safeTransfer(feeAddress,depositFee) (contract.sol#1583)
State variables written after the call(s):
- user.amount = user.amount.add(_amount).sub(depositFee) (contract.sol#1584)
- user.rewardDebt = user.amount.mul(pool.accSonicPerShare).div(1e12)
(contract.sol#1589)
Reentrancy in MasterChef.deposit(uint256,uint256,address) (contract.sol#1571-1591):
External calls:
- updatePool(_pid) (contract.sol#1574)
  - sonic.mint(devAddress,sonicReward.div(10)) (contract.sol#1564)
  - sonic.mint(address(this),sonicReward) (contract.sol#1565)
- sonicBoomReferral.recordReferral(msg.sender,_referrer) (contract.sol#1576)
- payOrLockuppendingSonic(_pid) (contract.sol#1578)
  - sonic.transfer(_to,sonicBal) (contract.sol#1655)
  - sonic.transfer(_to,_amount) (contract.sol#1657)
  - sonic.mint(referrer,commissionAmount) (contract.sol#1699)
  - sonicBoomReferral.recordReferralCommission(referrer,commissionAmount)
(contract.sol#1700)
- pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
(contract.sol#1580)
State variables written after the call(s):
- user.amount = user.amount.add(_amount) (contract.sol#1586)
Reentrancy in MasterChef.set(uint256,uint256,uint16,uint256,bool)
(contract.sol#1505-1515):
External calls:
- massUpdatePools() (contract.sol#1509)
  - sonic.mint(devAddress,sonicReward.div(10)) (contract.sol#1564)
  - sonic.mint(address(this),sonicReward) (contract.sol#1565)
State variables written after the call(s):
- poolInfo[_pid].allocPoint = _allocPoint (contract.sol#1512)
- poolInfo[_pid].depositFeeBP = _depositFeeBP (contract.sol#1513)
- poolInfo[_pid].harvestInterval = _harvestInterval (contract.sol#1514)
- totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint)
(contract.sol#1511)
```

```
Reentrancy in MasterChef.updateEmissionRate(uint256) (contract.sol#1675-1679):
	External calls:
	- massUpdatePools() (contract.sol#1676)
		- sonic.mint(devAddress,sonicReward.div(10)) (contract.sol#1564)
		- sonic.mint(address(this),sonicReward) (contract.sol#1565)
	State variables written after the call(s):
	- sonicPerBlock = _sonicPerBlock (contract.sol#1678)
Reentrancy in MasterChef.updatePool(uint256) (contract.sol#1552-1568):
	External calls:
	- sonic.mint(devAddress,sonicReward.div(10)) (contract.sol#1564)
	- sonic.mint(address(this),sonicReward) (contract.sol#1565)
	State variables written after the call(s):
	- pool.accSonicPerShare =
pool.accSonicPerShare.add(sonicReward.mul(1e12).div(lpSupply)) (contract.sol#1566)
	- pool.lastRewardBlock = block.number (contract.sol#1567)
Reentrancy in MasterChef.withdraw(uint256,uint256) (contract.sol#1594-1606):
	External calls:
	- updatePool(_pid) (contract.sol#1598)
		- sonic.mint(devAddress,sonicReward.div(10)) (contract.sol#1564)
		- sonic.mint(address(this),sonicReward) (contract.sol#1565)
	- payOrLockuppendingSonic(_pid) (contract.sol#1599)
		- sonic.transfer(_to,sonicBal) (contract.sol#1655)
		- sonic.transfer(_to,_amount) (contract.sol#1657)
		- sonic.mint(referrer,commissionAmount) (contract.sol#1699)
		- sonicBoomReferral.recordReferralCommission(referrer,commissionAmount)
(contract.sol#1700)
	State variables written after the call(s):
	- payOrLockuppendingSonic(_pid) (contract.sol#1599)
		- user.nextHarvestUntil = block.timestamp.add(pool.harvestInterval)
(contract.sol#1627)
		- user.rewardLockedUp = 0 (contract.sol#1637)
		- user.nextHarvestUntil = block.timestamp.add(pool.harvestInterval)
(contract.sol#1638)
		- user.rewardLockedUp = user.rewardLockedUp.add(pending) (contract.sol#1645)
	- user.amount = user.amount.sub(_amount) (contract.sol#1601)
Reentrancy in MasterChef.withdraw(uint256,uint256) (contract.sol#1594-1606):
	External calls:
	- updatePool(_pid) (contract.sol#1598)
		- sonic.mint(devAddress,sonicReward.div(10)) (contract.sol#1564)
		- sonic.mint(address(this),sonicReward) (contract.sol#1565)
	- payOrLockuppendingSonic(_pid) (contract.sol#1599)
```

```
- sonic.transfer(_to,sonicBal) (contract.sol#1655)
- sonic.transfer(_to,_amount) (contract.sol#1657)
- sonic.mint(referrer,commissionAmount) (contract.sol#1699)
- sonicBoomReferral.recordReferralCommission(referrer,commissionAmount)
(contract.sol#1700)
- pool.lpToken.safeTransfer(address(msg.sender),_amount) (contract.sol#1602)
State variables written after the call(s):
- user.rewardDebt = user.amount.mul(pool.accSonicPerShare).div(1e12)
(contract.sol#1604)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-1

BEP20.constructor(string,string).name (contract.sol#861) shadows:
- BEP20.name() (contract.sol#877-879) (function)
- IBEP20.name() (contract.sol#247) (function)
BEP20.constructor(string,string).symbol (contract.sol#861) shadows:
- BEP20.symbol() (contract.sol#891-893) (function)
- IBEP20.symbol() (contract.sol#242) (function)
BEP20.allowance(address,address).owner (contract.sol#925) shadows:
- Ownable.owner() (contract.sol#703-705) (function)
BEP20._approve(address,address,uint256).owner (contract.sol#1097) shadows:
- Ownable.owner() (contract.sol#703-705) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-
shadowing

MasterChef.add(uint256,IBEP20,uint16,uint256,bool) (contract.sol#1485-1502) should emit
an event for:
- totalAllocPoint = totalAllocPoint.add(_allocPoint) (contract.sol#1492)
MasterChef.set(uint256,uint256,uint16,uint256,bool) (contract.sol#1505-1515) should
emit an event for:
- totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint)
(contract.sol#1511)
MasterChef.setReferralCommissionRate(uint16) (contract.sol#1687-1690) should emit an
event for:
- referralCommissionRate = _referralCommissionRate (contract.sol#1689)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic

Reentrancy in MasterChef.deposit(uint256,uint256,address) (contract.sol#1571-1591):
External calls:
- updatePool(_pid) (contract.sol#1574)
```

```
- sonic.mint(devAddress,sonicReward.div(10)) (contract.sol#1564)
- sonic.mint(address(this),sonicReward) (contract.sol#1565)
- sonicBoomReferral.recordReferral(msg.sender,_referrer) (contract.sol#1576)
- payOrLockuppendingSonic(_pid) (contract.sol#1578)
- sonic.transfer(_to,sonicBal) (contract.sol#1655)
- sonic.transfer(_to,_amount) (contract.sol#1657)
- sonic.mint(referrer,commissionAmount) (contract.sol#1699)
- sonicBoomReferral.recordReferralCommission(referrer,commissionAmount)
(contract.sol#1700)
Event emitted after the call(s):
- ReferralCommissionPaid(_user,referrer,commissionAmount) (contract.sol#1701)
- payOrLockuppendingSonic(_pid) (contract.sol#1578)
- RewardLockedUp(msg.sender,_pid,pending) (contract.sol#1647)
- payOrLockuppendingSonic(_pid) (contract.sol#1578)
Reentrancy in MasterChef.deposit(uint256,uint256,address) (contract.sol#1571-1591):
External calls:
- updatePool(_pid) (contract.sol#1574)
- sonic.mint(devAddress,sonicReward.div(10)) (contract.sol#1564)
- sonic.mint(address(this),sonicReward) (contract.sol#1565)
- sonicBoomReferral.recordReferral(msg.sender,_referrer) (contract.sol#1576)
- payOrLockuppendingSonic(_pid) (contract.sol#1578)
- sonic.transfer(_to,sonicBal) (contract.sol#1655)
- sonic.transfer(_to,_amount) (contract.sol#1657)
- sonic.mint(referrer,commissionAmount) (contract.sol#1699)
- sonicBoomReferral.recordReferralCommission(referrer,commissionAmount)
(contract.sol#1700)
- pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
(contract.sol#1580)
- pool.lpToken.safeTransfer(feeAddress,depositFee) (contract.sol#1583)
Event emitted after the call(s):
- Deposit(msg.sender,_pid,_amount) (contract.sol#1590)
Reentrancy in MasterChef.emergencyWithdraw(uint256) (contract.sol#1609-1619):
External calls:
- pool.lpToken.safeTransfer(address(msg.sender),amount) (contract.sol#1617)
Event emitted after the call(s):
- EmergencyWithdraw(msg.sender,_pid,amount) (contract.sol#1618)
Reentrancy in MasterChef.payOrLockuppendingSonic(uint256) (contract.sol#1622-1649):
External calls:
- safeSonicTransfer(msg.sender,totalRewards) (contract.sol#1641)
- sonic.transfer(_to,sonicBal) (contract.sol#1655)
- sonic.transfer(_to,_amount) (contract.sol#1657)
```

```
- payReferralCommission(msg.sender,totalRewards) (contract.sol#1642)
  - sonic.mint(referrer,commissionAmount) (contract.sol#1699)
  - sonicBoomReferral.recordReferralCommission(referrer,commissionAmount)
(contract.sol#1700)
Event emitted after the call(s):
- ReferralCommissionPaid(_user,referrer,commissionAmount) (contract.sol#1701)
  - payReferralCommission(msg.sender,totalRewards) (contract.sol#1642)
Reentrancy in MasterChef.payReferralCommission(address,uint256)
(contract.sol#1693-1704):
External calls:
- sonic.mint(referrer,commissionAmount) (contract.sol#1699)
- sonicBoomReferral.recordReferralCommission(referrer,commissionAmount)
(contract.sol#1700)
Event emitted after the call(s):
- ReferralCommissionPaid(_user,referrer,commissionAmount) (contract.sol#1701)
Reentrancy in MasterChef.updateEmissionRate(uint256) (contract.sol#1675-1679):
External calls:
- massUpdatePools() (contract.sol#1676)
  - sonic.mint(devAddress,sonicReward.div(10)) (contract.sol#1564)
  - sonic.mint(address(this),sonicReward) (contract.sol#1565)
Event emitted after the call(s):
- EmissionRateUpdated(msg.sender,sonicPerBlock,_sonicPerBlock) (contract.sol#1677)
Reentrancy in MasterChef.withdraw(uint256,uint256) (contract.sol#1594-1606):
External calls:
- updatePool(_pid) (contract.sol#1598)
  - sonic.mint(devAddress,sonicReward.div(10)) (contract.sol#1564)
  - sonic.mint(address(this),sonicReward) (contract.sol#1565)
- payOrLockuppendingSonic(_pid) (contract.sol#1599)
  - sonic.transfer(_to,sonicBal) (contract.sol#1655)
  - sonic.transfer(_to,_amount) (contract.sol#1657)
  - sonic.mint(referrer,commissionAmount) (contract.sol#1699)
  - sonicBoomReferral.recordReferralCommission(referrer,commissionAmount)
(contract.sol#1700)
Event emitted after the call(s):
- ReferralCommissionPaid(_user,referrer,commissionAmount) (contract.sol#1701)
  - payOrLockuppendingSonic(_pid) (contract.sol#1599)
- RewardLockedUp(msg.sender,_pid,pending) (contract.sol#1647)
  - payOrLockuppendingSonic(_pid) (contract.sol#1599)
Reentrancy in MasterChef.withdraw(uint256,uint256) (contract.sol#1594-1606):
External calls:
- updatePool(_pid) (contract.sol#1598)
```

```
- sonic.mint(devAddress,sonicReward.div(10)) (contract.sol#1564)
- sonic.mint(address(this),sonicReward) (contract.sol#1565)
- payOrLockuppendingSonic(_pid) (contract.sol#1599)
- sonic.transfer(_to,sonicBal) (contract.sol#1655)
- sonic.transfer(_to,_amount) (contract.sol#1657)
- sonic.mint(referrer,commissionAmount) (contract.sol#1699)
- sonicBoomReferral.recordReferralCommission(referrer,commissionAmount)
(contract.sol#1700)
- pool.lpToken.safeTransfer(address(msg.sender),_amount) (contract.sol#1602)
Event emitted after the call(s):
- Withdraw(msg.sender,_pid,_amount) (contract.sol#1605)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3

SonicToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32)
(contract.sol#1202-1243) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(now <= expiry,SONIC::delegateBySig: signature expired)
(contract.sol#1241)
MasterChef.canHarvest(uint256,address) (contract.sol#1538-1541) uses timestamp for
comparisons
Dangerous comparisons:
- block.timestamp >= user.nextHarvestUntil (contract.sol#1540)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp

Address.isContract(address) (contract.sol#350-359) uses assembly
- INLINE ASM (contract.sol#357)
Address._verifyCallResult(bool,bytes,string) (contract.sol#495-512) uses assembly
- INLINE ASM (contract.sol#504-507)
SonicToken.getChainId() (contract.sol#1361-1365) uses assembly
- INLINE ASM (contract.sol#1363)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

MasterChef.nonDuplicated(IBEP20) (contract.sol#1478-1481) compares to a boolean
constant:
-require(bool,string)(poolExistence[_lpToken] == false,nonDuplicated: duplicated)
(contract.sol#1479)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-
equality
```

```
Different versions of Solidity is used:
- Version used: ['0.6.12', '>=0.4.0', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0', '^0.6.0']
- >=0.6.0<0.8.0 (contract.sol#9)
- >=0.4.0 (contract.sol#226)
- >=0.6.2<0.8.0 (contract.sol#327)
- ^0.6.0 (contract.sol#519)
- 0.6.12 (contract.sol#622)
- >=0.6.0<0.8.0 (contract.sol#645)
- >=0.6.0<0.8.0 (contract.sol#672)
- >=0.6.0<0.8.0 (contract.sol#742)
- >=0.4.0 (contract.sol#807)
- 0.6.12 (contract.sol#1126)
- 0.6.12 (contract.sol#1372)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-
pragma-directives-are-used

Address.functionCall(address,bytes) (contract.sol#403-405) is never used and should be
removed
Address.functionCallWithValue(address,bytes,uint256) (contract.sol#428-430) is never
used and should be removed
Address.functionDelegateCall(address,bytes) (contract.sol#477-479) is never used and
should be removed
Address.functionDelegateCall(address,bytes,string) (contract.sol#487-493) is never used
and should be removed
Address.functionStaticCall(address,bytes) (contract.sol#453-455) is never used and
should be removed
Address.functionStaticCall(address,bytes,string) (contract.sol#463-469) is never used
and should be removed
Address.sendValue(address,uint256) (contract.sol#377-383) is never used and should be
removed
BEP20._burn(address,uint256) (contract.sol#1075-1081) is never used and should be
removed
BEP20._burnFrom(address,uint256) (contract.sol#1114-1121) is never used and should be
removed
Context._msgData() (contract.sol#662-665) is never used and should be removed
SafeBEP20.safeApprove(IBEP20,address,uint256) (contract.sol#561-575) is never used and
should be removed
SafeBEP20.safeDecreaseAllowance(IBEP20,address,uint256) (contract.sol#586-596) is never
used and should be removed
SafeBEP20.safeIncreaseAllowance(IBEP20,address,uint256) (contract.sol#577-584) is never
used and should be removed
```

```
SafeMath.div(uint256,uint256,string) (contract.sol#196-199) is never used and should be
removed
SafeMath.mod(uint256,uint256) (contract.sol#158-161) is never used and should be
removed
SafeMath.mod(uint256,uint256,string) (contract.sol#216-219) is never used and should be
removed
SafeMath.tryAdd(uint256,uint256) (contract.sol#30-34) is never used and should be
removed
SafeMath.tryDiv(uint256,uint256) (contract.sol#66-69) is never used and should be
removed
SafeMath.tryMod(uint256,uint256) (contract.sol#76-79) is never used and should be
removed
SafeMath.tryMul(uint256,uint256) (contract.sol#51-59) is never used and should be
removed
SafeMath.trySub(uint256,uint256) (contract.sol#41-44) is never used and should be
removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version>=0.6.0<0.8.0 (contract.sol#9) is too complex
Pragma version>=0.4.0 (contract.sol#226) allows old versions
Pragma version>=0.6.2<0.8.0 (contract.sol#327) is too complex
Pragma version^0.6.0 (contract.sol#519) allows old versions
Pragma version>=0.6.0<0.8.0 (contract.sol#645) is too complex
Pragma version>=0.6.0<0.8.0 (contract.sol#672) is too complex
Pragma version>=0.6.0<0.8.0 (contract.sol#742) is too complex
Pragma version>=0.4.0 (contract.sol#807) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity

Low level call in Address.sendValue(address,uint256) (contract.sol#377-383):
 - (success) = recipient.call{value: amount}() (contract.sol#381)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string)
(contract.sol#438-445):
 - (success,returndata) = target.call{value: value}(data) (contract.sol#443)
Low level call in Address.functionStaticCall(address,bytes,string)
(contract.sol#463-469):
 - (success,returndata) = target.staticcall(data) (contract.sol#467)
Low level call in Address.functionDelegateCall(address,bytes,string)
(contract.sol#487-493):
 - (success,returndata) = target.delegatecall(data) (contract.sol#491)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls
```

```
Parameter SonicToken.mint(address,uint256)._to (contract.sol#1132) is not in mixedCase
Parameter SonicToken.mint(address,uint256)._amount (contract.sol#1132) is not in
mixedCase
Variable SonicToken._delegates (contract.sol#1144) is not in mixedCase
Parameter MasterChef.add(uint256,IBEP20,uint16,uint256,bool)._allocPoint
(contract.sol#1485) is not in mixedCase
Parameter MasterChef.add(uint256,IBEP20,uint16,uint256,bool)._lpToken
(contract.sol#1485) is not in mixedCase
Parameter MasterChef.add(uint256,IBEP20,uint16,uint256,bool)._depositFeeBP
(contract.sol#1485) is not in mixedCase
Parameter MasterChef.add(uint256,IBEP20,uint16,uint256,bool)._harvestInterval
(contract.sol#1485) is not in mixedCase
Parameter MasterChef.add(uint256,IBEP20,uint16,uint256,bool)._withUpdate
(contract.sol#1485) is not in mixedCase
Parameter MasterChef.set(uint256,uint256,uint16,uint256,bool)._pid (contract.sol#1505)
is not in mixedCase
Parameter MasterChef.set(uint256,uint256,uint16,uint256,bool)._allocPoint
(contract.sol#1505) is not in mixedCase
Parameter MasterChef.set(uint256,uint256,uint16,uint256,bool)._depositFeeBP
(contract.sol#1505) is not in mixedCase
Parameter MasterChef.set(uint256,uint256,uint16,uint256,bool)._harvestInterval
(contract.sol#1505) is not in mixedCase
Parameter MasterChef.set(uint256,uint256,uint16,uint256,bool)._withUpdate
(contract.sol#1505) is not in mixedCase
Parameter MasterChef.getMultiplier(uint256,uint256)._from (contract.sol#1518) is not in
mixedCase
Parameter MasterChef.getMultiplier(uint256,uint256)._to (contract.sol#1518) is not in
mixedCase
Parameter MasterChef.pendingSonic(uint256,address)._pid (contract.sol#1523) is not in
mixedCase
Parameter MasterChef.pendingSonic(uint256,address)._user (contract.sol#1523) is not in
mixedCase
Parameter MasterChef.canHarvest(uint256,address)._pid (contract.sol#1538) is not in
mixedCase
Parameter MasterChef.canHarvest(uint256,address)._user (contract.sol#1538) is not in
mixedCase
Parameter MasterChef.updatePool(uint256)._pid (contract.sol#1552) is not in mixedCase
Parameter MasterChef.deposit(uint256,uint256,address)._pid (contract.sol#1571) is not
in mixedCase
Parameter MasterChef.deposit(uint256,uint256,address)._amount (contract.sol#1571) is
```

```
not in mixedCase
Parameter MasterChef.deposit(uint256,uint256,address)._referrer (contract.sol#1571) is
not in mixedCase
Parameter MasterChef.withdraw(uint256,uint256)._pid (contract.sol#1594) is not in
mixedCase
Parameter MasterChef.withdraw(uint256,uint256)._amount (contract.sol#1594) is not in
mixedCase
Parameter MasterChef.emergencyWithdraw(uint256)._pid (contract.sol#1609) is not in
mixedCase
Parameter MasterChef.payOrLockuppendingSonic(uint256)._pid (contract.sol#1622) is not
in mixedCase
Parameter MasterChef.safeSonicTransfer(address,uint256)._to (contract.sol#1652) is not
in mixedCase
Parameter MasterChef.safeSonicTransfer(address,uint256)._amount (contract.sol#1652) is
not in mixedCase
Parameter MasterChef.setDevAddress(address)._devAddress (contract.sol#1662) is not in
mixedCase
Parameter MasterChef.setFeeAddress(address)._feeAddress (contract.sol#1668) is not in
mixedCase
Parameter MasterChef.updateEmissionRate(uint256)._sonicPerBlock (contract.sol#1675) is
not in mixedCase
Parameter MasterChef.setSonicBoomReferral(ISonicBoomReferral)._sonicBoomReferral
(contract.sol#1682) is not in mixedCase
Parameter MasterChef.setReferralCommissionRate(uint16)._referralCommissionRate
(contract.sol#1687) is not in mixedCase
Parameter MasterChef.payReferralCommission(address,uint256)._user (contract.sol#1693)
is not in mixedCase
Parameter MasterChef.payReferralCommission(address,uint256)._pending
(contract.sol#1693) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions

Redundant expression "this (contract.sol#663)" inContext (contract.sol#657-666)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-
statements

renounceOwnership() should be declared external:
- Ownable.renounceOwnership() (contract.sol#722-725)
transferOwnership(address) should be declared external:
- Ownable.transferOwnership(address) (contract.sol#731-735)
decimals() should be declared external:
```

```
- BEP20.decimals() (contract.sol#884-886)
symbol() should be declared external:
- BEP20.symbol() (contract.sol#891-893)
totalSupply() should be declared external:
- BEP20.totalSupply() (contract.sol#898-900)
transfer(address,uint256) should be declared external:
- BEP20.transfer(address,uint256) (contract.sol#917-920)
allowance(address,address) should be declared external:
- BEP20.allowance(address,address) (contract.sol#925-927)
approve(address,uint256) should be declared external:
- BEP20.approve(address,uint256) (contract.sol#936-939)
transferFrom(address,address,uint256) should be declared external:
- BEP20.transferFrom(address,address,uint256) (contract.sol#953-965)
increaseAllowance(address,uint256) should be declared external:
- BEP20.increaseAllowance(address,uint256) (contract.sol#979-982)
decreaseAllowance(address,uint256) should be declared external:
- BEP20.decreaseAllowance(address,uint256) (contract.sol#998-1005)
mint(uint256) should be declared external:
- BEP20.mint(uint256) (contract.sol#1015-1018)
mint(address,uint256) should be declared external:
- SonicToken.mint(address,uint256) (contract.sol#1132-1135)
add(uint256,IBEP20,uint16,uint256,bool) should be declared external:
- MasterChef.add(uint256,IBEP20,uint16,uint256,bool) (contract.sol#1485-1502)
set(uint256,uint256,uint16,uint256,bool) should be declared external:
- MasterChef.set(uint256,uint256,uint16,uint256,bool) (contract.sol#1505-1515)
deposit(uint256,uint256,address) should be declared external:
- MasterChef.deposit(uint256,uint256,address) (contract.sol#1571-1591)
withdraw(uint256,uint256) should be declared external:
- MasterChef.withdraw(uint256,uint256) (contract.sol#1594-1606)
emergencyWithdraw(uint256) should be declared external:
- MasterChef.emergencyWithdraw(uint256) (contract.sol#1609-1619)
setDevAddress(address) should be declared external:
- MasterChef.setDevAddress(address) (contract.sol#1662-1666)
setFeeAddress(address) should be declared external:
- MasterChef.setFeeAddress(address) (contract.sol#1668-1672)
updateEmissionRate(uint256) should be declared external:
- MasterChef.updateEmissionRate(uint256) (contract.sol#1675-1679)
setSonicBoomReferral(ISonicBoomReferral) should be declared external:
- MasterChef.setSonicBoomReferral(ISonicBoomReferral) (contract.sol#1682-1684)
setReferralCommissionRate(uint16) should be declared external:
- MasterChef.setReferralCommissionRate(uint16) (contract.sol#1687-1690)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external
. analyzed (11 contracts with 75 detectors), 130 result(s) found
```

0x Guard