Guard

Smart contracts security assessment

Final report

Tariff: Standard

Avalanche Rush

November 2021

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for the Avalanche Rush team. The project is a fork of Sushi's MasterChef contract modified for the Avalanche network and with fixes and added functionality (deposit and withdraw fees).

The ownership of the SonicToken was transferred to the SonicChef contract. The SonicChef ownership was transferred to the Timelock contract with a minimum delay of 24 hours.

An audit from Certik is available here. The current audit focuses only on the issues not found in the Certik report.

Name	Avalanche Rush
Audit date	2021-11-23 - 2021-11-23
Language	Solidity
Platform	Avalanche Network

Contracts checked

Name	Address
SonicToken	0x4Aca0ad6357b918e3d06BB1a0BCC403619177523
Referral	0x9b3eb72d7A4d743514EffB9a1641f7e751450Fa7
SonicChef	0x9178A7659701F81bdC82363b51567E33e488c16D
Timelock	0xEc969af5BEeB5DCc702CC53323aD7d1610626252

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Manually analyse smart contracts for security vulnerabilities
- Smart contracts' logic check

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	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Outdated Compiler Version	passed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

No issues were found

Medium severity issues

1. Function pendingSonic() may return wrong results (SonicChef)

The function pendingSonic calculates the SonicReward for the pool as Sonic token is minted with SonicPerSecond rate for the pool.

```
    function pendingSonic(uint256 _pid, address _user) external view returns (uint256)
{
        PoolInfo storage pool = poolInfo[_pid];
        UserInfo storage user = userInfo[_pid][_user];
        uint256 accSonicPerShare = pool.accSonicPerShare;
        uint256 lpSupply = pool.lpSupply;
        if (block.timestamp > pool.lastRewardTime && lpSupply != 0 &&
totalAllocPoint>0) {
            uint256 multiplier = getMultiplier(pool.lastRewardTime, block.timestamp);
            uint256 SonicReward =
multiplier.mul(SonicPerSecond).mul(pool.allocPoint).div(totalAllocPoint);
            accSonicPerShare =
accSonicPerShare.add(SonicReward.mul(1e18).div(lpSupply));
        }
        return user.amount.mul(accSonicPerShare).div(1e18).sub(user.rewardDebt);
    }
```

The actual mint rate is less because some tokens are minted for the dev address:

```
    function updatePool(uint256 _pid) public {
        ...
        uint256 devReward = SonicReward.div(10);
        uint256 reward = SonicReward.sub(devReward);
```

```
        Sonic.mint(devAddress, devReward);
        Sonic.mint(address(this), reward);
        ...
    }
```

The issue affects only a helper function for the frontend and does not impose any risks for the users.

Recommendation: As the contract is already deployed to the mainnet and can't be changed, tweak frontend calculations if needed. Also a helper contract can be written to simplify frontend calculations.

Low severity issues

No issues were found

Conclusion

Avalanche Rush SonicToken, Referral, SonicChef, Timelock contracts were audited. 1 medium severity issue was 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

Slither tool static code analysis result:

```
INFO:Detectors:
SonicChef.pendingSonic(uint256,address) (contracts/SonicChef.sol#159-170) performs a
multiplication on the result of a division:
        -SonicReward =
multiplier.mul(SonicPerSecond).mul(pool.allocPoint).div(totalAllocPoint) (contracts/
SonicChef.sol#166)
        -accSonicPerShare = accSonicPerShare.add(SonicReward.mul(1e18).div(lpSupply))
(contracts/SonicChef.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply
INFO:Detectors:
Reentrancy in SonicChef.add(uint256,IERC20,uint16,uint16,bool) (contracts/
SonicChef.sol#116-138):
        External calls:
        - massUpdatePools() (contracts/SonicChef.sol#123)
                - Sonic.mint(devAddress,devReward) (contracts/SonicChef.sol#195)
                - Sonic.mint(address(this),reward) (contracts/SonicChef.sol#196)
        State variables written after the call(s):
        - poolExistence[_lpToken] = true (contracts/SonicChef.sol#125)
        - poolInfo.push(PoolInfo(_lpToken,_allocPoint,lastRewardTime,0,_depositFeeBP,_wi
thdrawFeeBP,0)) (contracts/SonicChef.sol#126-135)
Reentrancy in SonicChef.deposit(uint256,uint256,address) (contracts/
SonicChef.sol#202-233):
        External calls:
        - updatePool(_pid) (contracts/SonicChef.sol#205)
                - Sonic.mint(devAddress,devReward) (contracts/SonicChef.sol#195)
                - Sonic.mint(address(this),reward) (contracts/SonicChef.sol#196)
        - referral.recordReferral(msg.sender,_referrer) (contracts/SonicChef.sol#207)
        - safeSonicTransfer(msg.sender,pending) (contracts/SonicChef.sol#212)
                - transferSuccess = Sonic.transfer(_to,SonicBal) (contracts/
SonicChef.sol#282)
                - transferSuccess = Sonic.transfer(_to,_amount) (contracts/
SonicChef.sol#284)
        - payReferralCommission(msg.sender,pending) (contracts/SonicChef.sol#213)
                - Sonic.mint(referrer,commissionAmount) (contracts/SonicChef.sol#324)
```

```
                - referral.recordReferralCommission(referrer,commissionAmount)
(contracts/SonicChef.sol#325)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
(contracts/SonicChef.sol#218)
        - pool.lpToken.safeTransfer(feeAddress,depositFee) (contracts/
SonicChef.sol#222)
        State variables written after the call(s):
        - pool.lpSupply = pool.lpSupply.add(finalDepositedAmount) (contracts/
SonicChef.sol#225)
        - user.amount = user.amount.add(finalDepositedAmount) (contracts/
SonicChef.sol#224)
Reentrancy in SonicChef.deposit(uint256,uint256,address) (contracts/
SonicChef.sol#202-233):
        External calls:
        - updatePool(_pid) (contracts/SonicChef.sol#205)
                - Sonic.mint(devAddress,devReward) (contracts/SonicChef.sol#195)
                - Sonic.mint(address(this),reward) (contracts/SonicChef.sol#196)
        - referral.recordReferral(msg.sender,_referrer) (contracts/SonicChef.sol#207)
        - safeSonicTransfer(msg.sender,pending) (contracts/SonicChef.sol#212)
                - transferSuccess = Sonic.transfer(_to,SonicBal) (contracts/
SonicChef.sol#282)
                - transferSuccess = Sonic.transfer(_to,_amount) (contracts/
SonicChef.sol#284)
        - payReferralCommission(msg.sender,pending) (contracts/SonicChef.sol#213)
                - Sonic.mint(referrer,commissionAmount) (contracts/SonicChef.sol#324)
                - referral.recordReferralCommission(referrer,commissionAmount)
(contracts/SonicChef.sol#325)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
(contracts/SonicChef.sol#218)
        State variables written after the call(s):
        - pool.lpSupply = pool.lpSupply.add(final_amount) (contracts/SonicChef.sol#228)
        - user.amount = user.amount.add(final_amount) (contracts/SonicChef.sol#227)
Reentrancy in SonicChef.set(uint256,uint256,uint16,uint16,bool) (contracts/
SonicChef.sol#141-151):
        External calls:
        - massUpdatePools() (contracts/SonicChef.sol#146)
                - Sonic.mint(devAddress,devReward) (contracts/SonicChef.sol#195)
                - Sonic.mint(address(this),reward) (contracts/SonicChef.sol#196)
        State variables written after the call(s):
        - poolInfo[_pid].allocPoint = _allocPoint (contracts/SonicChef.sol#148)
        - poolInfo[_pid].withdrawFeeBP = _withdrawFeeBP (contracts/SonicChef.sol#149)
```

```
Reentrancy in SonicChef.updateEmissionRate(uint256) (contracts/SonicChef.sol#303-308):
        External calls:
        - massUpdatePools() (contracts/SonicChef.sol#305)
                - Sonic.mint(devAddress,devReward) (contracts/SonicChef.sol#195)
                - Sonic.mint(address(this),reward) (contracts/SonicChef.sol#196)
        State variables written after the call(s):
        - SonicPerSecond = _SonicPerSecond (contracts/SonicChef.sol#306)
Reentrancy in SonicChef.updatePool(uint256) (contracts/SonicChef.sol#181-199):
        External calls:
        - Sonic.mint(devAddress,devReward) (contracts/SonicChef.sol#195)
        - Sonic.mint(address(this),reward) (contracts/SonicChef.sol#196)
        State variables written after the call(s):
        - pool.accSonicPerShare =
pool.accSonicPerShare.add(reward.mul(1e18).div(lpSupply)) (contracts/SonicChef.sol#197)
        - pool.lastRewardTime = block.timestamp (contracts/SonicChef.sol#198)
Reentrancy in SonicChef.withdraw(uint256,uint256) (contracts/SonicChef.sol#236-261):
        External calls:
        - updatePool(_pid) (contracts/SonicChef.sol#240)
                - Sonic.mint(devAddress,devReward) (contracts/SonicChef.sol#195)
                - Sonic.mint(address(this),reward) (contracts/SonicChef.sol#196)
        - safeSonicTransfer(msg.sender,pending) (contracts/SonicChef.sol#243)
                - transferSuccess = Sonic.transfer(_to,SonicBal) (contracts/
SonicChef.sol#282)
                - transferSuccess = Sonic.transfer(_to,_amount) (contracts/
SonicChef.sol#284)
        - payReferralCommission(msg.sender,pending) (contracts/SonicChef.sol#244)
                - Sonic.mint(referrer,commissionAmount) (contracts/SonicChef.sol#324)
                - referral.recordReferralCommission(referrer,commissionAmount)
(contracts/SonicChef.sol#325)
        State variables written after the call(s):
        - user.amount = user.amount.sub(_amount) (contracts/SonicChef.sol#247)
Reentrancy in SonicChef.withdraw(uint256,uint256) (contracts/SonicChef.sol#236-261):
        External calls:
        - updatePool(_pid) (contracts/SonicChef.sol#240)
                - Sonic.mint(devAddress,devReward) (contracts/SonicChef.sol#195)
                - Sonic.mint(address(this),reward) (contracts/SonicChef.sol#196)
        - safeSonicTransfer(msg.sender,pending) (contracts/SonicChef.sol#243)
                - transferSuccess = Sonic.transfer(_to,SonicBal) (contracts/
SonicChef.sol#282)
                - transferSuccess = Sonic.transfer(_to,_amount) (contracts/
SonicChef.sol#284)
```

```
        - payReferralCommission(msg.sender,pending) (contracts/SonicChef.sol#244)
                - Sonic.mint(referrer,commissionAmount) (contracts/SonicChef.sol#324)
                - referral.recordReferralCommission(referrer,commissionAmount)
(contracts/SonicChef.sol#325)
        - pool.lpToken.safeTransfer(feeAddress,withdrawFee) (contracts/
SonicChef.sol#251)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount.sub(withdrawFee))
(contracts/SonicChef.sol#252)
        State variables written after the call(s):
        - pool.lpSupply = pool.lpSupply.sub(_amount) (contracts/SonicChef.sol#253)
Reentrancy in SonicChef.withdraw(uint256,uint256) (contracts/SonicChef.sol#236-261):
        External calls:
        - updatePool(_pid) (contracts/SonicChef.sol#240)
                - Sonic.mint(devAddress,devReward) (contracts/SonicChef.sol#195)
                - Sonic.mint(address(this),reward) (contracts/SonicChef.sol#196)
        - safeSonicTransfer(msg.sender,pending) (contracts/SonicChef.sol#243)
                - transferSuccess = Sonic.transfer(_to,SonicBal) (contracts/
SonicChef.sol#282)
                - transferSuccess = Sonic.transfer(_to,_amount) (contracts/
SonicChef.sol#284)
        - payReferralCommission(msg.sender,pending) (contracts/SonicChef.sol#244)
                - Sonic.mint(referrer,commissionAmount) (contracts/SonicChef.sol#324)
                - referral.recordReferralCommission(referrer,commissionAmount)
(contracts/SonicChef.sol#325)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (contracts/
SonicChef.sol#255)
        State variables written after the call(s):
        - pool.lpSupply = pool.lpSupply.sub(_amount) (contracts/SonicChef.sol#256)
Reentrancy in SonicChef.withdraw(uint256,uint256) (contracts/SonicChef.sol#236-261):
        External calls:
        - updatePool(_pid) (contracts/SonicChef.sol#240)
                - Sonic.mint(devAddress,devReward) (contracts/SonicChef.sol#195)
                - Sonic.mint(address(this),reward) (contracts/SonicChef.sol#196)
        - safeSonicTransfer(msg.sender,pending) (contracts/SonicChef.sol#243)
                - transferSuccess = Sonic.transfer(_to,SonicBal) (contracts/
SonicChef.sol#282)
                - transferSuccess = Sonic.transfer(_to,_amount) (contracts/
SonicChef.sol#284)
        - payReferralCommission(msg.sender,pending) (contracts/SonicChef.sol#244)
                - Sonic.mint(referrer,commissionAmount) (contracts/SonicChef.sol#324)
                - referral.recordReferralCommission(referrer,commissionAmount)
```

```
(contracts/SonicChef.sol#325)
        - pool.lpToken.safeTransfer(feeAddress,withdrawFee) (contracts/
SonicChef.sol#251)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount.sub(withdrawFee))
(contracts/SonicChef.sol#252)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (contracts/
SonicChef.sol#255)
        State variables written after the call(s):
        - user.rewardDebt = user.amount.mul(pool.accSonicPerShare).div(1e18) (contracts/
SonicChef.sol#259)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-1
INFO:Detectors:
SonicChef.add(uint256,IERC20,uint16,uint16,bool) (contracts/SonicChef.sol#116-138)
ignores return value by _lpToken.balanceOf(address(this)) (contracts/SonicChef.sol#120)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
MockERC20.constructor(string,string,uint256).name (contracts/mocks/MockERC20.sol#10)
shadows:
        - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/
ERC20.sol#64-66) (function)
MockERC20.constructor(string,string,uint256).symbol (contracts/mocks/MockERC20.sol#10)
shadows:
        - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/
ERC20.sol#72-74) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-
shadowing
INFO:Detectors:
SonicChef.constructor(SonicToken,uint256,address,address,IReferral)._devAddress
(contracts/SonicChef.sol#91) lacks a zero-check on :
                - devAddress = _devAddress (contracts/SonicChef.sol#99)
SonicChef.constructor(SonicToken,uint256,address,address,IReferral)._feeAddress
(contracts/SonicChef.sol#92) lacks a zero-check on :
                - feeAddress = _feeAddress (contracts/SonicChef.sol#100)
Timelock.constructor(address,uint256).admin_ (contracts/Timelock.sol#43) lacks a zero-
check on :
                - admin = admin_ (contracts/Timelock.sol#47)
Timelock.setPendingAdmin(address).pendingAdmin_ (contracts/Timelock.sol#72) lacks a
zero-check on :
                - pendingAdmin = pendingAdmin_ (contracts/Timelock.sol#80)
Timelock.executeTransaction(address,uint256,string,bytes,uint256).target (contracts/
```

```
Timelock.sol#105) lacks a zero-check on :
                - (success,returnData) = target.call.value(value)(callData) (contracts/
Timelock.sol#124)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
address-validation
INFO:Detectors:
Multicall.aggregate(Multicall.Call[]) (contracts/Multicall.sol#16-24) has external
calls inside a loop: (success,ret) = calls[i].target.call(calls[i].callData) (contracts/
Multicall.sol#20)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-
a-loop
INFO:Detectors:
Reentrancy in SonicChef.add(uint256,IERC20,uint16,uint16,bool) (contracts/
SonicChef.sol#116-138):
        External calls:
        - massUpdatePools() (contracts/SonicChef.sol#123)
                - Sonic.mint(devAddress,devReward) (contracts/SonicChef.sol#195)
                - Sonic.mint(address(this),reward) (contracts/SonicChef.sol#196)
        Event emitted after the call(s):
        - Add(_allocPoint,_lpToken,_depositFeeBP,_withdrawFeeBP) (contracts/
SonicChef.sol#137)
Reentrancy in SonicChef.deposit(uint256,uint256,address) (contracts/
SonicChef.sol#202-233):
        External calls:
        - updatePool(_pid) (contracts/SonicChef.sol#205)
                - Sonic.mint(devAddress,devReward) (contracts/SonicChef.sol#195)
                - Sonic.mint(address(this),reward) (contracts/SonicChef.sol#196)
        - referral.recordReferral(msg.sender,_referrer) (contracts/SonicChef.sol#207)
        - safeSonicTransfer(msg.sender,pending) (contracts/SonicChef.sol#212)
                - transferSuccess = Sonic.transfer(_to,SonicBal) (contracts/
SonicChef.sol#282)
                - transferSuccess = Sonic.transfer(_to,_amount) (contracts/
SonicChef.sol#284)
        - payReferralCommission(msg.sender,pending) (contracts/SonicChef.sol#213)
                - Sonic.mint(referrer,commissionAmount) (contracts/SonicChef.sol#324)
                - referral.recordReferralCommission(referrer,commissionAmount)
(contracts/SonicChef.sol#325)
        Event emitted after the call(s):
        - ReferralCommissionPaid(_user,referrer,commissionAmount) (contracts/
SonicChef.sol#326)
                - payReferralCommission(msg.sender,pending) (contracts/
SonicChef.sol#213)
```

```
Reentrancy in SonicChef.deposit(uint256,uint256,address) (contracts/
SonicChef.sol#202-233):
        External calls:
        - updatePool(_pid) (contracts/SonicChef.sol#205)
                - Sonic.mint(devAddress,devReward) (contracts/SonicChef.sol#195)
                - Sonic.mint(address(this),reward) (contracts/SonicChef.sol#196)
        - referral.recordReferral(msg.sender,_referrer) (contracts/SonicChef.sol#207)
        - safeSonicTransfer(msg.sender,pending) (contracts/SonicChef.sol#212)
                - transferSuccess = Sonic.transfer(_to,SonicBal) (contracts/
SonicChef.sol#282)
                - transferSuccess = Sonic.transfer(_to,_amount) (contracts/
SonicChef.sol#284)
        - payReferralCommission(msg.sender,pending) (contracts/SonicChef.sol#213)
                - Sonic.mint(referrer,commissionAmount) (contracts/SonicChef.sol#324)
                - referral.recordReferralCommission(referrer,commissionAmount)
(contracts/SonicChef.sol#325)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
(contracts/SonicChef.sol#218)
        - pool.lpToken.safeTransfer(feeAddress,depositFee) (contracts/
SonicChef.sol#222)
        Event emitted after the call(s):
        - Deposit(msg.sender,_pid,_amount) (contracts/SonicChef.sol#232)
Reentrancy in SonicChef.emergencyWithdraw(uint256) (contracts/SonicChef.sol#264-275):
        External calls:
        - pool.lpToken.safeTransfer(feeAddress,withdrawFee) (contracts/
SonicChef.sol#272)
        - pool.lpToken.safeTransfer(address(msg.sender),amount.sub(withdrawFee))
(contracts/SonicChef.sol#273)
        Event emitted after the call(s):
        - EmergencyWithdraw(msg.sender,_pid,amount) (contracts/SonicChef.sol#274)
Reentrancy in Timelock.executeTransaction(address,uint256,string,bytes,uint256)
(contracts/Timelock.sol#105-130):
        External calls:
        - (success,returnData) = target.call.value(value)(callData) (contracts/
Timelock.sol#124)
        Event emitted after the call(s):
        - ExecuteTransaction(txHash,target,value,signature,data,eta) (contracts/
Timelock.sol#127)
Reentrancy in SonicChef.payReferralCommission(address,uint256) (contracts/
SonicChef.sol#318-329):
        External calls:
```

```
        - Sonic.mint(referrer,commissionAmount) (contracts/SonicChef.sol#324)
        - referral.recordReferralCommission(referrer,commissionAmount) (contracts/
SonicChef.sol#325)
        Event emitted after the call(s):
        - ReferralCommissionPaid(_user,referrer,commissionAmount) (contracts/
SonicChef.sol#326)
Reentrancy in SonicChef.set(uint256,uint256,uint16,uint16,bool) (contracts/
SonicChef.sol#141-151):
        External calls:
        - massUpdatePools() (contracts/SonicChef.sol#146)
                - Sonic.mint(devAddress,devReward) (contracts/SonicChef.sol#195)
                - Sonic.mint(address(this),reward) (contracts/SonicChef.sol#196)
        Event emitted after the call(s):
        - Set(_pid,_allocPoint,_depositFeeBP,_withdrawFeeBP) (contracts/
SonicChef.sol#150)
Reentrancy in SonicChef.updateEmissionRate(uint256) (contracts/SonicChef.sol#303-308):
        External calls:
        - massUpdatePools() (contracts/SonicChef.sol#305)
                - Sonic.mint(devAddress,devReward) (contracts/SonicChef.sol#195)
                - Sonic.mint(address(this),reward) (contracts/SonicChef.sol#196)
        Event emitted after the call(s):
        - UpdateEmissionRate(msg.sender,_SonicPerSecond) (contracts/SonicChef.sol#307)
Reentrancy in SonicChef.withdraw(uint256,uint256) (contracts/SonicChef.sol#236-261):
        External calls:
        - updatePool(_pid) (contracts/SonicChef.sol#240)
                - Sonic.mint(devAddress,devReward) (contracts/SonicChef.sol#195)
                - Sonic.mint(address(this),reward) (contracts/SonicChef.sol#196)
        - safeSonicTransfer(msg.sender,pending) (contracts/SonicChef.sol#243)
                - transferSuccess = Sonic.transfer(_to,SonicBal) (contracts/
SonicChef.sol#282)
                - transferSuccess = Sonic.transfer(_to,_amount) (contracts/
SonicChef.sol#284)
        - payReferralCommission(msg.sender,pending) (contracts/SonicChef.sol#244)
                - Sonic.mint(referrer,commissionAmount) (contracts/SonicChef.sol#324)
                - referral.recordReferralCommission(referrer,commissionAmount)
(contracts/SonicChef.sol#325)
        Event emitted after the call(s):
        - ReferralCommissionPaid(_user,referrer,commissionAmount) (contracts/
SonicChef.sol#326)
                - payReferralCommission(msg.sender,pending) (contracts/
SonicChef.sol#244)
```

```
Reentrancy in SonicChef.withdraw(uint256,uint256) (contracts/SonicChef.sol#236-261):
        External calls:
        - updatePool(_pid) (contracts/SonicChef.sol#240)
                - Sonic.mint(devAddress,devReward) (contracts/SonicChef.sol#195)
                - Sonic.mint(address(this),reward) (contracts/SonicChef.sol#196)
        - safeSonicTransfer(msg.sender,pending) (contracts/SonicChef.sol#243)
                - transferSuccess = Sonic.transfer(_to,SonicBal) (contracts/
SonicChef.sol#282)
                - transferSuccess = Sonic.transfer(_to,_amount) (contracts/
SonicChef.sol#284)
        - payReferralCommission(msg.sender,pending) (contracts/SonicChef.sol#244)
                - Sonic.mint(referrer,commissionAmount) (contracts/SonicChef.sol#324)
                - referral.recordReferralCommission(referrer,commissionAmount)
(contracts/SonicChef.sol#325)
        - pool.lpToken.safeTransfer(feeAddress,withdrawFee) (contracts/
SonicChef.sol#251)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount.sub(withdrawFee))
(contracts/SonicChef.sol#252)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (contracts/
SonicChef.sol#255)
        Event emitted after the call(s):
        - Withdraw(msg.sender,_pid,_amount) (contracts/SonicChef.sol#260)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3
INFO:Detectors:
SonicChef.add(uint256,IERC20,uint16,uint16,bool) (contracts/SonicChef.sol#116-138) uses
timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > startTime (contracts/SonicChef.sol#119)
SonicChef.pendingSonic(uint256,address) (contracts/SonicChef.sol#159-170) uses
timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > pool.lastRewardTime && lpSupply != 0 && totalAllocPoint > 0
(contracts/SonicChef.sol#164)
SonicChef.massUpdatePools() (contracts/SonicChef.sol#173-178) uses timestamp for
comparisons
        Dangerous comparisons:
        - pid < length (contracts/SonicChef.sol#175)
SonicChef.updatePool(uint256) (contracts/SonicChef.sol#181-199) uses timestamp for
comparisons
        Dangerous comparisons:
```

```
        - block.timestamp <= pool.lastRewardTime (contracts/SonicChef.sol#183)
SonicChef.updateStartTime(uint256) (contracts/SonicChef.sol#332-343) uses timestamp for
comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp < startTime,cannot change start
timestamp if farm has already started) (contracts/SonicChef.sol#333)
        - require(bool,string)(block.timestamp < _newStartTime,cannot set start
timestamp in the past) (contracts/SonicChef.sol#334)
Timelock.queueTransaction(address,uint256,string,bytes,uint256) (contracts/
Timelock.sol#85-94) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(eta >=
getBlockTimestamp().add(delay),Timelock::queueTransaction: Estimated execution block
must satisfy delay.) (contracts/Timelock.sol#87)
Timelock.executeTransaction(address,uint256,string,bytes,uint256) (contracts/
Timelock.sol#105-130) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(getBlockTimestamp() >= eta,Timelock::executeTransaction:
Transaction hasn't surpassed time lock.) (contracts/Timelock.sol#110)
        - require(bool,string)(getBlockTimestamp() <=
eta.add(GRACE_PERIOD),Timelock::executeTransaction: Transaction is stale.) (contracts/
Timelock.sol#111)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp
INFO:Detectors:
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/
Address.sol#26-35) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/
utils/Address.sol#171-188) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#180-183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
SonicChef.nonDuplicated(IERC20) (contracts/SonicChef.sol#110-113) compares to a boolean
constant:
        -require(bool,string)(poolExistence[_lpToken] == false,nonDuplicated:
duplicated) (contracts/SonicChef.sol#111)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-
equality
INFO:Detectors:
Different versions of Solidity is used:
```

```
        - Version used: ['0.6.12', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0']
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
SafeERC20.sol#3)
        - >=0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
        - >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/
ReentrancyGuard.sol#3)
        - 0.6.12 (contracts/Referral.sol#3)
        - 0.6.12 (contracts/SonicChef.sol#3)
        - 0.6.12 (contracts/SonicToken.sol#3)
        - 0.6.12 (contracts/Timelock.sol#16)
        - 0.6.12 (contracts/interfaces/IReferral.sol#3)
        - 0.6.12 (contracts/mocks/MockERC20.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-
pragma-directives-are-used
INFO:Detectors:
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3)
is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
ERC20.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
IERC20.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
SafeERC20.sol#3) is too complex
Pragma version>=0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
is too complex
Pragma version>=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/
ReentrancyGuard.sol#3) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity
INFO:Detectors:
Pragma version>=0.5.0 (contracts/interfaces/IPangolinFactory.sol#3) allows old versions
solc-0.8.0 is not recommended for deployment
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity
INFO:Detectors:
Pragma version>=0.5.0 (contracts/Multicall.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/
contracts/utils/Address.sol#53-59):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/
contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string)
(node_modules/@openzeppelin/contracts/utils/Address.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (node_modules/
@openzeppelin/contracts/utils/Address.sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/
@openzeppelin/contracts/utils/Address.sol#139-145):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#143)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/
@openzeppelin/contracts/utils/Address.sol#163-169):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/
contracts/utils/Address.sol#167)
Low level call in Timelock.executeTransaction(address,uint256,string,bytes,uint256)
(contracts/Timelock.sol#105-130):
        - (success,returnData) = target.call.value(value)(callData) (contracts/
Timelock.sol#124)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls
INFO:Detectors:
Low level call in Multicall.aggregate(Multicall.Call[]) (contracts/
Multicall.sol#16-24):
        - (success,ret) = calls[i].target.call(calls[i].callData) (contracts/
Multicall.sol#20)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls
INFO:Detectors:
Parameter Referral.recordReferral(address,address)._user (contracts/Referral.sol#26) is
not in mixedCase
Parameter Referral.recordReferral(address,address)._referrer (contracts/
Referral.sol#26) is not in mixedCase
```

Parameter Referral.recordReferralCommission(address,uint256)._referrer (contracts/Referral.sol#38) is not in mixedCase
Parameter Referral.recordReferralCommission(address,uint256)._commission (contracts/Referral.sol#38) is not in mixedCase
Parameter Referral.getReferrer(address)._user (contracts/Referral.sol#46) is not in mixedCase
Parameter Referral.updateOperator(address,bool)._operator (contracts/Referral.sol#51) is not in mixedCase
Parameter Referral.updateOperator(address,bool)._status (contracts/Referral.sol#51) is not in mixedCase
Parameter SonicChef.add(uint256,IERC20,uint16,uint16,bool)._allocPoint (contracts/SonicChef.sol#116) is not in mixedCase
Parameter SonicChef.add(uint256,IERC20,uint16,uint16,bool)._lpToken (contracts/SonicChef.sol#116) is not in mixedCase
Parameter SonicChef.add(uint256,IERC20,uint16,uint16,bool)._depositFeeBP (contracts/SonicChef.sol#116) is not in mixedCase
Parameter SonicChef.add(uint256,IERC20,uint16,uint16,bool)._withdrawFeeBP (contracts/SonicChef.sol#116) is not in mixedCase
Parameter SonicChef.add(uint256,IERC20,uint16,uint16,bool)._withUpdate (contracts/SonicChef.sol#116) is not in mixedCase
Parameter SonicChef.set(uint256,uint256,uint16,uint16,bool)._pid (contracts/SonicChef.sol#141) is not in mixedCase
Parameter SonicChef.set(uint256,uint256,uint16,uint16,bool)._allocPoint (contracts/SonicChef.sol#141) is not in mixedCase
Parameter SonicChef.set(uint256,uint256,uint16,uint16,bool)._depositFeeBP (contracts/SonicChef.sol#141) is not in mixedCase
Parameter SonicChef.set(uint256,uint256,uint16,uint16,bool)._withdrawFeeBP (contracts/SonicChef.sol#141) is not in mixedCase
Parameter SonicChef.set(uint256,uint256,uint16,uint16,bool)._withUpdate (contracts/SonicChef.sol#141) is not in mixedCase
Parameter SonicChef.getMultiplier(uint256,uint256)._from (contracts/SonicChef.sol#154) is not in mixedCase
Parameter SonicChef.getMultiplier(uint256,uint256)._to (contracts/SonicChef.sol#154) is not in mixedCase
Parameter SonicChef.pendingSonic(uint256,address)._pid (contracts/SonicChef.sol#159) is not in mixedCase
Parameter SonicChef.pendingSonic(uint256,address)._user (contracts/SonicChef.sol#159) is not in mixedCase
Parameter SonicChef.updatePool(uint256)._pid (contracts/SonicChef.sol#181) is not in mixedCase
Parameter SonicChef.deposit(uint256,uint256,address)._pid (contracts/SonicChef.sol#202)

```
is not in mixedCase
Parameter SonicChef.deposit(uint256,uint256,address)._amount (contracts/
SonicChef.sol#202) is not in mixedCase
Parameter SonicChef.deposit(uint256,uint256,address)._referrer (contracts/
SonicChef.sol#202) is not in mixedCase
Parameter SonicChef.withdraw(uint256,uint256)._pid (contracts/SonicChef.sol#236) is not
in mixedCase
Parameter SonicChef.withdraw(uint256,uint256)._amount (contracts/SonicChef.sol#236) is
not in mixedCase
Parameter SonicChef.emergencyWithdraw(uint256)._pid (contracts/SonicChef.sol#264) is
not in mixedCase
Parameter SonicChef.safeSonicTransfer(address,uint256)._to (contracts/
SonicChef.sol#278) is not in mixedCase
Parameter SonicChef.safeSonicTransfer(address,uint256)._amount (contracts/
SonicChef.sol#278) is not in mixedCase
Parameter SonicChef.setDevAddress(address)._devAddress (contracts/SonicChef.sol#290) is
not in mixedCase
Parameter SonicChef.setFeeAddress(address)._feeAddress (contracts/SonicChef.sol#296) is
not in mixedCase
Parameter SonicChef.updateEmissionRate(uint256)._SonicPerSecond (contracts/
SonicChef.sol#303) is not in mixedCase
Parameter SonicChef.setReferralCommissionRate(uint16)._referralCommissionRate
(contracts/SonicChef.sol#311) is not in mixedCase
Parameter SonicChef.payReferralCommission(address,uint256)._user (contracts/
SonicChef.sol#318) is not in mixedCase
Parameter SonicChef.payReferralCommission(address,uint256)._pending (contracts/
SonicChef.sol#318) is not in mixedCase
Parameter SonicChef.updateStartTime(uint256)._newStartTime (contracts/
SonicChef.sol#332) is not in mixedCase
Variable SonicChef.Sonic (contracts/SonicChef.sol#51) is not in mixedCase
Variable SonicChef.SonicPerSecond (contracts/SonicChef.sol#56) is not in mixedCase
Variable Timelock.admin_initialized (contracts/Timelock.sol#38) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (node_modules/@openzeppelin/contracts/utils/Context.sol#21)"
inContext (node_modules/@openzeppelin/contracts/utils/Context.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-
statements
INFO:Detectors:
renounceOwnership() should be declared external:
```

```
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/
Ownable.sol#54-57)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/
access/Ownable.sol#63-67)
name() should be declared external:
        - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/
ERC20.sol#64-66)
symbol() should be declared external:
        - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/
ERC20.sol#72-74)
decimals() should be declared external:
        - ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/
ERC20.sol#89-91)
totalSupply() should be declared external:
        - ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/
ERC20.sol#96-98)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/
ERC20.sol#103-105)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/
ERC20/ERC20.sol#115-118)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (node_modules/@openzeppelin/contracts/token/
ERC20/ERC20.sol#123-125)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/
ERC20/ERC20.sol#134-137)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/
contracts/token/ERC20/ERC20.sol#152-156)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/
contracts/token/ERC20/ERC20.sol#170-173)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/
contracts/token/ERC20/ERC20.sol#189-192)
setDelay(uint256) should be declared external:
        - Timelock.setDelay(uint256) (contracts/Timelock.sol#55-62)
acceptAdmin() should be declared external:
```

```
        - Timelock.acceptAdmin() (contracts/Timelock.sol#64-70)
setPendingAdmin(address) should be declared external:
        - Timelock.setPendingAdmin(address) (contracts/Timelock.sol#72-83)
queueTransaction(address,uint256,string,bytes,uint256) should be declared external:
        - Timelock.queueTransaction(address,uint256,string,bytes,uint256) (contracts/
Timelock.sol#85-94)
cancelTransaction(address,uint256,string,bytes,uint256) should be declared external:
        - Timelock.cancelTransaction(address,uint256,string,bytes,uint256) (contracts/
Timelock.sol#96-103)
executeTransaction(address,uint256,string,bytes,uint256) should be declared external:
        - Timelock.executeTransaction(address,uint256,string,bytes,uint256) (contracts/
Timelock.sol#105-130)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external
INFO:Detectors:
aggregate(Multicall.Call[]) should be declared external:
        - Multicall.aggregate(Multicall.Call[]) (contracts/Multicall.sol#16-24)
getEthBalance(address) should be declared external:
        - Multicall.getEthBalance(address) (contracts/Multicall.sol#26-28)
getBlockHash(uint256) should be declared external:
        - Multicall.getBlockHash(uint256) (contracts/Multicall.sol#29-31)
getLastBlockHash() should be declared external:
        - Multicall.getLastBlockHash() (contracts/Multicall.sol#32-34)
getCurrentBlockTimestamp() should be declared external:
        - Multicall.getCurrentBlockTimestamp() (contracts/Multicall.sol#35-37)
getCurrentBlockDifficulty() should be declared external:
        - Multicall.getCurrentBlockDifficulty() (contracts/Multicall.sol#38-40)
getCurrentBlockGasLimit() should be declared external:
        - Multicall.getCurrentBlockGasLimit() (contracts/Multicall.sol#41-43)
getCurrentBlockCoinbase() should be declared external:
        - Multicall.getCurrentBlockCoinbase() (contracts/Multicall.sol#44-46)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external
INFO:Slither:. analyzed (16 contracts with 75 detectors), 126 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github
integration
```

0x Guard