0x Guard

Smart contracts security assessment

Final report

Tariff: Standard

Nextgen masterchef

March 2022

0xguard.com

hello@0xguard.com

Contents

1. Introduction	3
2. Contracts checked	3
3. Procedure	3
4. Known vulnerabilities checked	4
5. Classification of issue severity	5
6. Issues	5
7. Conclusion	8
8. Disclaimer	9
9. Slither output	10

Introduction

The report has been prepared for the Nextgen team.

The audited code is deployed at address 0xE97d4a9CA8c66Fcd9FC0Dd10197DE805F493772D.

The audited contract is a Masterchef contract with the possibility of adding commissions to the pool deposit. The Masterchef has a pool of his own token, the allocPoint of which are equal to the sum of the remaining allocPoint of the pools divided by 3. BEP20 interface is implemented with the use of OpenZeppelin libraries, which is considered the best practice.

Name	Nextgen masterchef
Audit date	2022-03-11 - 2022-03-11
Language	Solidity
Platform	Polygon Network

Contracts checked

Name	Address
MasterChef	0xE97d4a9CA8c66Fcd9FC0Dd10197DE805F493772D

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	Not 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Outdated Compiler Version	passed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. Engine token is not burnt during emergency withdrawal (MasterChef)

An attacker can use the enterStaking() function and then make an emergency withdrawal uses emergencyWithdraw(). This will allow attacker to mint as many engine tokens as he wants.

Recommendation: Burn tokens in the emergencyWithdraw function

Nextget team response: The Engine receipt token has no liquidity, has no value and cannot be redeemed in any way. This nullifies this exploit, there is no benefit to having the extra tokens, as they will also not carry any weight in governance.

2. Exploit on voting mechanisms (MasterChef)

The delegates in the NextGen and Engine tokens are not transferred. There is a known attack on the delegation mechanism. This allows an attacker to mint any voting power.

Nextgen team response: We are looking into creating a system that will automatically flag and nullify any votes caused by these attacks. All Governance votes will be manually checked until this system is in place to ensure a fair voting system.

3. Owner can update ngenPerBlock (MasterChef)

Open access to the setNgenPerBlockEmissionRate() function can lead to fraudulent activity with the owner or a compromised owner account. If ngenPerBlock will be equal to 0, the block rewards will be 0.

Recommendation: Use a multisig wallet and put it behind a Timelock contract by giving it owner rights. After this the severity of the issue may be lowered.

Nextgen team response: The team has full control over this and to add remove and modify content as per community recommendations, Time lock will be implemented once community is fit to govern itself. there is no multisig system on the fuse network. The team is doxed and don't see this as an issue as the personal ramifications of any nefarious activities would be devastating.

4. Owner can update multiplier (MasterChef)

Open access to the updateMultiplier() function can lead to fraudulent activity with the owner or a compromised owner account. If Owner changes the BONUS_MULTIPLIER in updateMultiplier() function to 0, the block rewards will be 0.

Recommendation: Use a multisig wallet and put it behind a Timelock contract by giving it owner rights. After this the severity of the issue may be lowered.

Nextgen team response: The team has full control over this and to add remove and modify content as per community recommendations, Time lock will be implemented once community is fit to govern itself. there is no multisig system on the fuse network. The team is doxed and don't see this as an issue as the personal ramifications of any nefarious activities would be devastating.

Medium severity issues

1. Tokens with fees on transfers are not supported (MasterChef)

If a token with commission on transfers is added, an attacker can use a known exploit to mint and sell minted tokens.

Recommendation: Check actual amount of deposited tokens by checking balance before and after token transfers in the deposit() function.

2. nonDuplicated modifer is not working (MasterChef)

poolExistence variable never changes, poolExistence[_lpToken] == false will always be true.

Low severity issues

1. Gas optimization (MasterChef)

The function updateStakingPool() uses a for() to loop through the pools to get points. The best solution would be to simply subtract poolInfo[0].allocPoint from totalAllocPoint

Conclusion

Nextgen masterchef MasterChef contract was audited. 4 high, 2 medium, 1 low severity 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.

Slither output

Engine.safeNgenTransfer(address,uint256) (masterchef.sol#1274-1281) ignores return value by ngen.transfer(_to,ngenBal) (masterchef.sol#1277)

Engine.safeNgenTransfer(address,uint256) (masterchef.sol#1274-1281) ignores return value by ngen.transfer(_to,_amount) (masterchef.sol#1279)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

MasterChef.poolExistence (masterchef.sol#1604) is never initialized. It is used in:

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

MasterChef.pendingNgen(uint256,address) (masterchef.sol#1661-1672) performs a multiplication on the result of a division:

-ngenReward = multiplier.mul(ngenPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (masterchef.sol#1668)

-accNgenPerShare = accNgenPerShare.add(ngenReward.mul(1e12).div(lpSupply)) (masterchef.sol#1669)

MasterChef.updatePool(uint256) (masterchef.sol#1684-1700) performs a multiplication on the result of a division:

-ngenReward = multiplier.mul(ngenPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (masterchef.sol#1695)

-pool.accNgenPerShare = pool.accNgenPerShare.add(ngenReward.mul(1e12).div(lpSupply)) (masterchef.sol#1698)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

NextGen._writeCheckpoint(address,uint32,uint256,uint256) (masterchef.sol#1217-1235) uses a dangerous strict equality:

- nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (masterchef.sol#1227)

Engine._writeCheckpoint(address,uint32,uint256,uint256) (masterchef.sol#1481-1499) uses a dangerous strict equality:

- nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (masterchef.sol#1491)

MasterChef.updatePool(uint256) (masterchef.sol#1684-1700) uses a dangerous strict equality:

- lpSupply == 0 (masterchef.sol#1690)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in MasterChef.add(uint256,IBEP20,uint16,bool) (masterchef.sol#1612-1626):

External calls:

- massUpdatePools() (masterchef.sol#1614)

- ngen.mint(devaddr,ngenReward.div(10)) (masterchef.sol#1696)

- ngen.mint(address(engine),ngenReward) (masterchef.sol#1697)

State variables written after the call(s):

- poolInfo.push(PoolInfo(_lpToken,_allocPoint,lastRewardBlock,0,_depositFeeBP)) (masterchef.sol#1618-1624)

- updateStakingPool() (masterchef.sol#1625)

    - poolInfo[0].allocPoint = points (masterchef.sol#1651)

- totalAllocPoint = totalAllocPoint.add(_allocPoint) (masterchef.sol#1617)

- updateStakingPool() (masterchef.sol#1625)

    - totalAllocPoint = totalAllocPoint.sub(poolInfo[0].allocPoint).add(points) (masterchef.sol#1650)

Reentrancy in MasterChef.deposit(uint256,uint256) (masterchef.sol#1703-1728):

External calls:

- updatePool(_pid) (masterchef.sol#1709)

    - ngen.mint(devaddr,ngenReward.div(10)) (masterchef.sol#1696)

    - ngen.mint(address(engine),ngenReward) (masterchef.sol#1697)

- safeNgenTransfer(msg.sender,pending) (masterchef.sol#1713)

    - engine.safeNgenTransfer(_to,_amount) (masterchef.sol#1804)

- pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (masterchef.sol#1717)

- pool.lpToken.safeTransfer(feeAddress,depositFee) (masterchef.sol#1720)

State variables written after the call(s):

- user.amount = user.amount.add(_amount).sub(depositFee) (masterchef.sol#1721)

Reentrancy in MasterChef.deposit(uint256,uint256) (masterchef.sol#1703-1728):

External calls:

- updatePool(_pid) (masterchef.sol#1709)

  - ngen.mint(devaddr,ngenReward.div(10)) (masterchef.sol#1696)

  - ngen.mint(address(engine),ngenReward) (masterchef.sol#1697)

- safeNgenTransfer(msg.sender,pending) (masterchef.sol#1713)

  - engine.safeNgenTransfer(_to,_amount) (masterchef.sol#1804)

- pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (masterchef.sol#1717)

State variables written after the call(s):

- user.amount = user.amount.add(_amount) (masterchef.sol#1723)

Reentrancy in MasterChef.emergencyWithdraw(uint256) (masterchef.sol#1793-1800):

External calls:

- pool.lpToken.safeTransfer(address(msg.sender),user.amount) (masterchef.sol#1796)

State variables written after the call(s):

- user.amount = 0 (masterchef.sol#1798)

- user.rewardDebt = 0 (masterchef.sol#1799)

Reentrancy in MasterChef.enterStaking(uint256) (masterchef.sol#1752-1770):

External calls:

- updatePool(0) (masterchef.sol#1755)

- ngen.mint(devaddr,ngenReward.div(10)) (masterchef.sol#1696)

- ngen.mint(address(engine),ngenReward) (masterchef.sol#1697)

- safeNgenTransfer(msg.sender,pending) (masterchef.sol#1759)

- engine.safeNgenTransfer(_to,_amount) (masterchef.sol#1804)

- pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (masterchef.sol#1763)

State variables written after the call(s):

- user.amount = user.amount.add(_amount) (masterchef.sol#1764)

- user.rewardDebt = user.amount.mul(pool.accNgenPerShare).div(1e12) (masterchef.sol#1766)

Reentrancy in MasterChef.leaveStaking(uint256) (masterchef.sol#1773-1790):

External calls:

- updatePool(0) (masterchef.sol#1777)

- ngen.mint(devaddr,ngenReward.div(10)) (masterchef.sol#1696)

- ngen.mint(address(engine),ngenReward) (masterchef.sol#1697)

- safeNgenTransfer(msg.sender,pending) (masterchef.sol#1780)

- engine.safeNgenTransfer(_to,_amount) (masterchef.sol#1804)

State variables written after the call(s):

- user.amount = user.amount.sub(_amount) (masterchef.sol#1783)

Reentrancy in MasterChef.leaveStaking(uint256) (masterchef.sol#1773-1790):

External calls:

- updatePool(0) (masterchef.sol#1777)

    - ngen.mint(devaddr,ngenReward.div(10)) (masterchef.sol#1696)

    - ngen.mint(address(engine),ngenReward) (masterchef.sol#1697)

- safeNgenTransfer(msg.sender,pending) (masterchef.sol#1780)

    - engine.safeNgenTransfer(_to,_amount) (masterchef.sol#1804)

- pool.lpToken.safeTransfer(address(msg.sender),_amount) (masterchef.sol#1784)

State variables written after the call(s):

- user.rewardDebt = user.amount.mul(pool.accNgenPerShare).div(1e12) (masterchef.sol#1786)

Reentrancy in MasterChef.set(uint256,uint256,uint16,bool) (masterchef.sol#1629-1640):

External calls:

- massUpdatePools() (masterchef.sol#1631)

    - ngen.mint(devaddr,ngenReward.div(10)) (masterchef.sol#1696)

    - ngen.mint(address(engine),ngenReward) (masterchef.sol#1697)

State variables written after the call(s):

- poolInfo[_pid].allocPoint = _allocPoint (masterchef.sol#1634)

- poolInfo[_pid].depositFeeBP = _depositFeeBP (masterchef.sol#1635)

- updateStakingPool() (masterchef.sol#1638)

- poolInfo[0].allocPoint = points (masterchef.sol#1651)

- totalAllocPoint = totalAllocPoint.sub(prevAllocPoint).add(_allocPoint) (masterchef.sol#1637)

- updateStakingPool() (masterchef.sol#1638)

- totalAllocPoint = totalAllocPoint.sub(poolInfo[0].allocPoint).add(points) (masterchef.sol#1650)

Reentrancy in MasterChef.setNgenPerBlockEmissionRate(uint256) (masterchef.sol#1813-1818):

External calls:

- massUpdatePools() (masterchef.sol#1816)

- ngen.mint(devaddr,ngenReward.div(10)) (masterchef.sol#1696)

- ngen.mint(address(engine),ngenReward) (masterchef.sol#1697)

State variables written after the call(s):

- ngenPerBlock = _ngenPerBlock (masterchef.sol#1817)

Reentrancy in MasterChef.updatePool(uint256) (masterchef.sol#1684-1700):

External calls:

- ngen.mint(devaddr,ngenReward.div(10)) (masterchef.sol#1696)

- ngen.mint(address(engine),ngenReward) (masterchef.sol#1697)

State variables written after the call(s):

- pool.accNgenPerShare = pool.accNgenPerShare.add(ngenReward.mul(1e12).div(lpSupply)) (masterchef.sol#1698)

```
    - pool.lastRewardBlock = block.number (masterchef.sol#1699)

Reentrancy in MasterChef.withdraw(uint256,uint256) (masterchef.sol#1731-1749):

    External calls:

    - updatePool(_pid) (masterchef.sol#1738)

        - ngen.mint(devaddr,ngenReward.div(10)) (masterchef.sol#1696)

        - ngen.mint(address(engine),ngenReward) (masterchef.sol#1697)

    - safeNgenTransfer(msg.sender,pending) (masterchef.sol#1741)

        - engine.safeNgenTransfer(_to,_amount) (masterchef.sol#1804)

    State variables written after the call(s):

    - user.amount = user.amount.sub(_amount) (masterchef.sol#1744)

Reentrancy in MasterChef.withdraw(uint256,uint256) (masterchef.sol#1731-1749):

    External calls:

    - updatePool(_pid) (masterchef.sol#1738)

        - ngen.mint(devaddr,ngenReward.div(10)) (masterchef.sol#1696)

        - ngen.mint(address(engine),ngenReward) (masterchef.sol#1697)

    - safeNgenTransfer(msg.sender,pending) (masterchef.sol#1741)

        - engine.safeNgenTransfer(_to,_amount) (masterchef.sol#1804)

    - pool.lpToken.safeTransfer(address(msg.sender),_amount) (masterchef.sol#1745)
```

State variables written after the call(s):

- user.rewardDebt = user.amount.mul(pool.accNgenPerShare).div(1e12) (masterchef.sol#1747)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

BEP20.constructor(string,string).name (masterchef.sol#747) shadows:

- BEP20.name() (masterchef.sol#763-765) (function)

- IBEP20.name() (masterchef.sol#211) (function)

BEP20.constructor(string,string).symbol (masterchef.sol#747) shadows:

- BEP20.symbol() (masterchef.sol#777-779) (function)

- IBEP20.symbol() (masterchef.sol#206) (function)

BEP20.allowance(address,address).owner (masterchef.sol#811) shadows:

- Ownable.owner() (masterchef.sol#597-599) (function)

BEP20._approve(address,address,uint256).owner (masterchef.sol#983) shadows:

- Ownable.owner() (masterchef.sol#597-599) (function)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

MasterChef.updateMultiplier(uint256) (masterchef.sol#1597-1599) should emit an event for:

- BONUS_MULTIPLIER = multiplierNumber (masterchef.sol#1598)

MasterChef.add(uint256,IBEP20,uint16,bool) (masterchef.sol#1612-1626) should emit an event for:

- totalAllocPoint = totalAllocPoint.add(_allocPoint) (masterchef.sol#1617)

MasterChef.set(uint256,uint256,uint16,bool) (masterchef.sol#1629-1640) should emit an event for:

- totalAllocPoint = totalAllocPoint.sub(prevAllocPoint).add(_allocPoint) (masterchef.sol#1637)

MasterChef.setNgenPerBlockEmissionRate(uint256) (masterchef.sol#1813-1818) should emit an event for:

- ngenPerBlock = _ngenPerBlock (masterchef.sol#1817)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

MasterChef.constructor(NextGen,Engine,address,address,uint256,uint256)._devaddr (masterchef.sol#1572) lacks a zero-check on :

- devaddr = _devaddr (masterchef.sol#1579)

MasterChef.constructor(NextGen,Engine,address,address,uint256,uint256)._feeAddress (masterchef.sol#1573) lacks a zero-check on :

- feeAddress = _feeAddress (masterchef.sol#1580)

MasterChef.dev(address)._devaddr (masterchef.sol#1808) lacks a zero-check on :

- devaddr = _devaddr (masterchef.sol#1810)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

MasterChef.updatePool(uint256) (masterchef.sol#1684-1700) has external calls inside a loop: lpSupply = pool.lpToken.balanceOf(address(this)) (masterchef.sol#1689)

MasterChef.updatePool(uint256) (masterchef.sol#1684-1700) has external calls inside a loop: ngen.mint(devaddr,ngenReward.div(10)) (masterchef.sol#1696)

MasterChef.updatePool(uint256) (masterchef.sol#1684-1700) has external calls inside a loop:

```
ngen.mint(address(engine),ngenReward) (masterchef.sol#1697)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in MasterChef.deposit(uint256,uint256) (masterchef.sol#1703-1728):

    External calls:

    - updatePool(_pid) (masterchef.sol#1709)

        - ngen.mint(devaddr,ngenReward.div(10)) (masterchef.sol#1696)

        - ngen.mint(address(engine),ngenReward) (masterchef.sol#1697)

    - safeNgenTransfer(msg.sender,pending) (masterchef.sol#1713)

        - engine.safeNgenTransfer(_to,_amount) (masterchef.sol#1804)

    - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
(masterchef.sol#1717)

    - pool.lpToken.safeTransfer(feeAddress,depositFee) (masterchef.sol#1720)

    Event emitted after the call(s):

    - Deposit(msg.sender,_pid,_amount) (masterchef.sol#1727)

Reentrancy in MasterChef.emergencyWithdraw(uint256) (masterchef.sol#1793-1800):

    External calls:

    - pool.lpToken.safeTransfer(address(msg.sender),user.amount) (masterchef.sol#1796)

    Event emitted after the call(s):

    - EmergencyWithdraw(msg.sender,_pid,user.amount) (masterchef.sol#1797)
```

Reentrancy in MasterChef.enterStaking(uint256) (masterchef.sol#1752-1770):

External calls:

- updatePool(0) (masterchef.sol#1755)

    - ngen.mint(devaddr,ngenReward.div(10)) (masterchef.sol#1696)

    - ngen.mint(address(engine),ngenReward) (masterchef.sol#1697)

- safeNgenTransfer(msg.sender,pending) (masterchef.sol#1759)

    - engine.safeNgenTransfer(_to,_amount) (masterchef.sol#1804)

- pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (masterchef.sol#1763)

- engine.mint(msg.sender,_amount) (masterchef.sol#1768)

Event emitted after the call(s):

- Deposit(msg.sender,0,_amount) (masterchef.sol#1769)

Reentrancy in MasterChef.leaveStaking(uint256) (masterchef.sol#1773-1790):

External calls:

- updatePool(0) (masterchef.sol#1777)

    - ngen.mint(devaddr,ngenReward.div(10)) (masterchef.sol#1696)

    - ngen.mint(address(engine),ngenReward) (masterchef.sol#1697)

- safeNgenTransfer(msg.sender,pending) (masterchef.sol#1780)

    - engine.safeNgenTransfer(_to,_amount) (masterchef.sol#1804)

- pool.lpToken.safeTransfer(address(msg.sender),_amount) (masterchef.sol#1784)

- engine.burn(msg.sender,_amount) (masterchef.sol#1788)

Event emitted after the call(s):

- Withdraw(msg.sender,0,_amount) (masterchef.sol#1789)

Reentrancy in MasterChef.withdraw(uint256,uint256) (masterchef.sol#1731-1749):

External calls:

- updatePool(_pid) (masterchef.sol#1738)

    - ngen.mint(devaddr,ngenReward.div(10)) (masterchef.sol#1696)

    - ngen.mint(address(engine),ngenReward) (masterchef.sol#1697)

- safeNgenTransfer(msg.sender,pending) (masterchef.sol#1741)

    - engine.safeNgenTransfer(_to,_amount) (masterchef.sol#1804)

- pool.lpToken.safeTransfer(address(msg.sender),_amount) (masterchef.sol#1745)

Event emitted after the call(s):

- Withdraw(msg.sender,_pid,_amount) (masterchef.sol#1748)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

NextGen.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (masterchef.sol#1083-1124) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(now <= expiry,NGEN::delegateBySig: signature expired)

(masterchef.sol#1122)

Engine.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (masterchef.sol#1347-1388) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(now <= expiry,nGen::delegateBySig: signature expired) (masterchef.sol#1386)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.isContract(address) (masterchef.sol#309-320) uses assembly

- INLINE ASM (masterchef.sol#316-318)

Address._functionCallWithValue(address,bytes,uint256,string) (masterchef.sol#417-443) uses assembly

- INLINE ASM (masterchef.sol#435-438)

NextGen.getChainId() (masterchef.sol#1242-1246) uses assembly

- INLINE ASM (masterchef.sol#1244)

Engine.getChainId() (masterchef.sol#1506-1510) uses assembly

- INLINE ASM (masterchef.sol#1508)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

MasterChef.nonDuplicated(IBEP20) (masterchef.sol#1605-1608) compares to a boolean constant:

-require(bool,string)(poolExistence[_lpToken] == false,nonDuplicated: duplicated) (masterchef.sol#1606)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Address.functionCall(address,bytes) (masterchef.sol#364-366) is never used and should be removed

Address.functionCallWithValue(address,bytes,uint256) (masterchef.sol#393-399) is never used and should be removed

Address.functionCallWithValue(address,bytes,uint256,string) (masterchef.sol#407-415) is never used and should be removed

Address.sendValue(address,uint256) (masterchef.sol#338-344) is never used and should be removed

BEP20._burnFrom(address,uint256) (masterchef.sol#1000-1007) is never used and should be removed

Context._msgData() (masterchef.sol#561-564) is never used and should be removed

SafeBEP20.safeApprove(IBEP20,address,uint256) (masterchef.sol#484-498) is never used and should be removed

SafeBEP20.safeDecreaseAllowance(IBEP20,address,uint256) (masterchef.sol#509-519) is never used and should be removed

SafeBEP20.safeIncreaseAllowance(IBEP20,address,uint256) (masterchef.sol#500-507) is never used and should be removed

SafeMath.min(uint256,uint256) (masterchef.sol#172-174) is never used and should be removed

SafeMath.mod(uint256,uint256) (masterchef.sol#147-149) is never used and should be removed

SafeMath.mod(uint256,uint256,string) (masterchef.sol#163-170) is never used and should be removed

SafeMath.sqrt(uint256) (masterchef.sol#177-188) is never used and should be removed

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Low level call in Address.sendValue(address,uint256) (masterchef.sol#338-344):

- (success) = recipient.call{value: amount}() (masterchef.sol#342)

Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (masterchef.sol#417-443):

- (success,returndata) = target.call{value: weiValue}(data) (masterchef.sol#426)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter NextGen.mint(address,uint256)._to (masterchef.sol#1014) is not in mixedCase

Parameter NextGen.mint(address,uint256)._amount (masterchef.sol#1014) is not in mixedCase

Variable NextGen._delegates (masterchef.sol#1025) is not in mixedCase

Parameter Engine.mint(address,uint256)._to (masterchef.sol#1253) is not in mixedCase

Parameter Engine.mint(address,uint256)._amount (masterchef.sol#1253) is not in mixedCase

Parameter Engine.burn(address,uint256)._from (masterchef.sol#1258) is not in mixedCase

Parameter Engine.burn(address,uint256)._amount (masterchef.sol#1258) is not in mixedCase

Parameter Engine.safeNgenTransfer(address,uint256)._to (masterchef.sol#1274) is not in mixedCase

Parameter Engine.safeNgenTransfer(address,uint256)._amount (masterchef.sol#1274) is not in mixedCase

Variable Engine._delegates (masterchef.sol#1289) is not in mixedCase

Parameter MasterChef.add(uint256,IBEP20,uint16,bool)._allocPoint (masterchef.sol#1612) is not in mixedCase

Parameter MasterChef.add(uint256,IBEP20,uint16,bool)._lpToken (masterchef.sol#1612) is not in mixedCase

Parameter MasterChef.add(uint256,IBEP20,uint16,bool)._depositFeeBP (masterchef.sol#1612) is not in mixedCase

Parameter MasterChef.add(uint256,IBEP20,uint16,bool)._withUpdate (masterchef.sol#1612) is not in mixedCase

Parameter MasterChef.set(uint256,uint256,uint16,bool)._pid (masterchef.sol#1629) is not in mixedCase

Parameter MasterChef.set(uint256,uint256,uint16,bool)._allocPoint (masterchef.sol#1629) is not in mixedCase

Parameter MasterChef.set(uint256,uint256,uint16,bool)._depositFeeBP (masterchef.sol#1629) is not in mixedCase

Parameter MasterChef.set(uint256,uint256,uint16,bool)._withUpdate (masterchef.sol#1629) is not in mixedCase

Parameter MasterChef.getMultiplier(uint256,uint256)._from (masterchef.sol#1656) is not in mixedCase

Parameter MasterChef.getMultiplier(uint256,uint256)._to (masterchef.sol#1656) is not in mixedCase

Parameter MasterChef.pendingNgen(uint256,address)._pid (masterchef.sol#1661) is not in mixedCase

Parameter MasterChef.pendingNgen(uint256,address)._user (masterchef.sol#1661) is not in mixedCase

Parameter MasterChef.updatePool(uint256)._pid (masterchef.sol#1684) is not in mixedCase

Parameter MasterChef.deposit(uint256,uint256)._pid (masterchef.sol#1703) is not in mixedCase

Parameter MasterChef.deposit(uint256,uint256)._amount (masterchef.sol#1703) is not in mixedCase

Parameter MasterChef.withdraw(uint256,uint256)._pid (masterchef.sol#1731) is not in mixedCase

Parameter MasterChef.withdraw(uint256,uint256)._amount (masterchef.sol#1731) is not in mixedCase

Parameter MasterChef.enterStaking(uint256)._amount (masterchef.sol#1752) is not in mixedCase

Parameter MasterChef.leaveStaking(uint256)._amount (masterchef.sol#1773) is not in mixedCase

Parameter MasterChef.emergencyWithdraw(uint256)._pid (masterchef.sol#1793) is not in mixedCase

Parameter MasterChef.safeNgenTransfer(address,uint256)._to (masterchef.sol#1803) is not in mixedCase

Parameter MasterChef.safeNgenTransfer(address,uint256)._amount (masterchef.sol#1803) is not in mixedCase

Parameter MasterChef.dev(address)._devaddr (masterchef.sol#1808) is not in mixedCase

Parameter MasterChef.setNgenPerBlockEmissionRate(uint256)._ngenPerBlock (masterchef.sol#1813) is not in mixedCase

Parameter MasterChef.updateStartBlock(uint256)._startBlock (masterchef.sol#1819) is not in mixedCase

Variable MasterChef.BONUS_MULTIPLIER (masterchef.sol#1553) is not in mixedCase

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (masterchef.sol#562)" inContext (masterchef.sol#552-565)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

renounceOwnership() should be declared external:

- Ownable.renounceOwnership() (masterchef.sol#616-619)

transferOwnership(address) should be declared external:

- Ownable.transferOwnership(address) (masterchef.sol#625-627)

decimals() should be declared external:

- BEP20.decimals() (masterchef.sol#770-772)

symbol() should be declared external:

- BEP20.symbol() (masterchef.sol#777-779)

totalSupply() should be declared external:

- BEP20.totalSupply() (masterchef.sol#784-786)

transfer(address,uint256) should be declared external:

- BEP20.transfer(address,uint256) (masterchef.sol#803-806)

allowance(address,address) should be declared external:

- BEP20.allowance(address,address) (masterchef.sol#811-813)

approve(address,uint256) should be declared external:

- BEP20.approve(address,uint256) (masterchef.sol#822-825)

transferFrom(address,address,uint256) should be declared external:

- BEP20.transferFrom(address,address,uint256) (masterchef.sol#839-851)

increaseAllowance(address,uint256) should be declared external:

- BEP20.increaseAllowance(address,uint256) (masterchef.sol#865-868)

decreaseAllowance(address,uint256) should be declared external:

- BEP20.decreaseAllowance(address,uint256) (masterchef.sol#884-891)

mint(uint256) should be declared external:

- BEP20.mint(uint256) (masterchef.sol#901-904)

mint(address,uint256) should be declared external:

- NextGen.mint(address,uint256) (masterchef.sol#1014-1017)

mint(address,uint256) should be declared external:

- Engine.mint(address,uint256) (masterchef.sol#1253-1256)

burn(address,uint256) should be declared external:

- Engine.burn(address,uint256) (masterchef.sol#1258-1261)

safeNgenTransfer(address,uint256) should be declared external:

- Engine.safeNgenTransfer(address,uint256) (masterchef.sol#1274-1281)

updateMultiplier(uint256) should be declared external:

- MasterChef.updateMultiplier(uint256) (masterchef.sol#1597-1599)

add(uint256,IBEP20,uint16,bool) should be declared external:

- MasterChef.add(uint256,IBEP20,uint16,bool) (masterchef.sol#1612-1626)

set(uint256,uint256,uint16,bool) should be declared external:

- MasterChef.set(uint256,uint256,uint16,bool) (masterchef.sol#1629-1640)

deposit(uint256,uint256) should be declared external:

- MasterChef.deposit(uint256,uint256) (masterchef.sol#1703-1728)

withdraw(uint256,uint256) should be declared external:

- MasterChef.withdraw(uint256,uint256) (masterchef.sol#1731-1749)

enterStaking(uint256) should be declared external:

- MasterChef.enterStaking(uint256) (masterchef.sol#1752-1770)

leaveStaking(uint256) should be declared external:

- MasterChef.leaveStaking(uint256) (masterchef.sol#1773-1790)

emergencyWithdraw(uint256) should be declared external:

- MasterChef.emergencyWithdraw(uint256) (masterchef.sol#1793-1800)

dev(address) should be declared external:

- MasterChef.dev(address) (masterchef.sol#1808-1811)

setNgenPerBlockEmissionRate(uint256) should be declared external:

- MasterChef.setNgenPerBlockEmissionRate(uint256) (masterchef.sol#1813-1818)

updateStartBlock(uint256) should be declared external:

- MasterChef.updateStartBlock(uint256) (masterchef.sol#1819-1829)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

Guard