Guard

Smart contracts security assessment

Final report

Tariff: Standard

OpenXSwap

April 2023

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for **OpenXSwap**.

The OpenXSwap is the fork of the Sushiswap project with some changes. It allows to swap and stake tokens.

Name	OpenXSwap
Audit date	2023-04-21 - 2023-04-25
Language	Solidity
Platform	Optimism Network

Contracts checked

Name	Address
ContractDeployer	
xPool	0xCD476505861BDe63942eF0BceBC2fb2538e46765
MasterChefV2O	0x237aeF9e106f35406ba435d865Ab151E2bA82d7B
OpenXGov	0x2513486f18eeE1498D7b6281f668B955181Dd0D9
OpenXMaker	0x1D5a5061fA9bd120576aA8062856BF161C94089b
OpenX	0xc3864f98f2a61A7cAeb95b039D031b4E2f55e0e9
TokenPaymentManager	0xdcdA0f3A9ffAFC2f9DE4380f62733B16C6FF51b6,0x90a8E9B9DffB04c3742787085B343e1FB8a71ED9,0x24d547055a881B37096A21cC2F01cC6DE36988d3
UniswapV2ERC20	
UniswapV2Factory	0xf3C7978Ddd70B4158b53e897f980093183cA5c52
UniswapV2Pair	
UniswapV2Router02	0x744776F27080b584D447A780ba260c435f3aE7d5
opx	0x46f21fDa29F1339e0aB543763FF683D399e393eC

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 analyze 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	not 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Outdated Compiler Version	passed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

No issues were found

Low severity issues

1. Variable default visibility (ContractDeployer)
Status: Open

The variable `owner` has default visibility. Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.

2. Gas optimization (ContractDeployer)
Status: Open

Visibility of the function `deployCtrct()` can be declared as 'external' to save gas.

3. Emergency Withdraw Flow (MasterChefV2O)
Status: Open

The `emergencyWithdraw()` function allows withdrawing tokens without caring about rewards. But in the current implementation rewards are still requested with external call to the `_rewarder` contract.

```
    function emergencyWithdraw(uint256 pid, address to) public {
        ...
        IRewarder _rewarder = rewarder[pid];
        if (address(_rewarder) != address(0)) {
```

```
            _rewarder.onSushiReward(pid, msg.sender, to, 0, 0);
        }
        ...
    }
```

Recommendation: We recommend removing the reward claiming functionality from the `emergencyWithdraw()` or putting it into the try-catch block.

4. Gas optimization (MasterChefV2O)
Status: Open

Visibility of the functions `add()`, `set()`, `rewardsPerSecond()`, `deposit()`, `withdraw()`, `claimRewards()`,`withdrawAndHarvest()`, `emergencyWithdraw()` can be declared as 'external' to save gas.

5. Variable default visibility (MasterChefV2O)
Status: Open

The mapping `isExistant` has default visibility. Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.

6. Gas optimization (OpenXGov)
Status: Open

1. The variables `OpenX`, `dummyToken` can be declared as 'immutable'.

2. Visibility of the functions `changeSnapshotAdmin()`, `snapshot()`, `enter()`, `init()`, `leave()` can be declared as 'external' to save gas.

7. Variable default visibility (OpenXGov)
Status: Open

The variable `dummyToken` has default visibility. Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.

8. Lack of non-zero check (OpenXGov)
Status: Open

We recommend adding non-zero checks for the input address parameters of the contract constructor.

9. Gas optimization (OpenXMaker)
Status: Open

Visibility of the functions `setAdmin()`, `burnX()` can be declared as 'external' to save gas.

10. Gas optimization (OpenX)
Status: Open

Visibility of the function `burn()` can be declared as 'external' to save gas.

11. Lack of non-zero check (TokenPaymentManager)
Status: Open

We recommend adding non-zero checks for the input address parameters of the contract constructor.

12. Gas optimization (TokenPaymentManager)
Status: Open

1. The variable `initialFundingAmount` can be declared as 'immutable'.

1. The variable `totalTime` can be declared as 'constant'.

2. Visibility of the functions `sendPayment()`, `init()`, `recoverToken()`, `recoverETH()` can be declared as 'external' to save gas.

13. Gas optimization (opx)
Status: Open

1. The variable `bondedToken` can be declared as 'immutable'.

2. Visibility of the functions `changeAdmin()`, `disableDeposit()`, `snapshot()`, `deposit()` can be declared as 'external' to save gas.

Conclusion

OpenXSwap ContractDeployer, xPool, MasterChefV2O, OpenXGov, OpenXMaker, OpenX, TokenPaymentManager, UniswapV2ERC20, UniswapV2Factory, UniswapV2Pair, UniswapV2Router02, opx contracts were audited. 13 low severity issues were found.

We strongly recommend writing unit tests to have extensive coverage of the codebase minimize the possibility of bugs and ensure that everything works as expected.

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.
OxGuard retains exclusive publishing rights for the results of this audit on its website and social networks.

Slither output

```
UniswapV2Pair._update(uint256,uint256,uint112,uint112) (contracts/uniswapv2/
UniswapV2Pair.sol#86-99) uses a weak PRNG: "blockTimestamp = uint32(block.timestamp % 2
** 32) (contracts/uniswapv2/UniswapV2Pair.sol#88)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

BaseBoringBatchable.batch(bytes[],bool) (contracts/MasterChefV2O.sol#326-335) has
delegatecall inside a loop in a payable function: (success,result) =
address(this).delegatecall(calls[i]) (contracts/MasterChefV2O.sol#330)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#payable-
functions-using-delegatecall-inside-a-loop

TokenPaymentManager.sendPayment() (contracts/TreasuryFundManager.sol#251-259) ignores
return value by Token.transfer(owner,amount) (contracts/TreasuryFundManager.sol#258)
TokenPaymentManager.init() (contracts/TreasuryFundManager.sol#266-270) ignores return
value by Token.transferFrom(msg.sender,address(this),initialFundingAmount) (contracts/
TreasuryFundManager.sol#269)
TokenPaymentManager.recoverToken(address) (contracts/TreasuryFundManager.sol#274-278)
ignores return value by IERC20(tokenAddr).transfer(owner,bal) (contracts/
TreasuryFundManager.sol#277)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-
transfer

OpenXGov.enter(uint256) (contracts/OpenXbar.sol#55-73) ignores return value by
OpenX.transferFrom(msg.sender,address(this),_amount) (contracts/OpenXbar.sol#72)
OpenXGov.init(IMC) (contracts/OpenXbar.sol#79-91) ignores return value by
OpenX.transferFrom(msg.sender,address(this),285889 * 10 ** 18) (contracts/
OpenXbar.sol#83)
OpenXGov.init(IMC) (contracts/OpenXbar.sol#79-91) ignores return value by
dummyToken.transferFrom(msg.sender,address(this),balance) (contracts/OpenXbar.sol#88)
OpenXGov.leave(uint256) (contracts/OpenXbar.sol#95-104) ignores return value by
OpenX.transfer(msg.sender,what) (contracts/OpenXbar.sol#103)
OpenXMaker._convert(address,address) (contracts/OpenXmaker.sol#109-123) ignores return
value by IERC20(address(pair)).transfer(address(pair),pair.balanceOf(address(this)))
(contracts/OpenXmaker.sol#116)
OpenXMaker._convertStep(address,address,uint256,uint256) (contracts/
OpenXmaker.sol#128-176) ignores return value by
IERC20(openx).transfer(bar,amount.div(3)) (contracts/OpenXmaker.sol#134)
```

```
OpenXMaker._convertStep(address,address,uint256,uint256) (contracts/
OpenXmaker.sol#128-176) ignores return value by
IERC20(openx).transfer(bar,amount0.div(3)) (contracts/OpenXmaker.sol#145)
OpenXMaker._convertStep(address,address,uint256,uint256) (contracts/
OpenXmaker.sol#128-176) ignores return value by
IERC20(openx).transfer(bar,amount1.div(3)) (contracts/OpenXmaker.sol#150)
OpenXMaker._swap(address,address,uint256,address) (contracts/OpenXmaker.sol#181-202)
ignores return value by IERC20(fromToken).transfer(address(pair),amountIn) (contracts/
OpenXmaker.sol#193)
OpenXMaker._swap(address,address,uint256,address) (contracts/OpenXmaker.sol#181-202)
ignores return value by IERC20(fromToken).transfer(address(pair),amountIn) (contracts/
OpenXmaker.sol#198)
OpenXMaker._toOPENX(address,uint256) (contracts/OpenXmaker.sol#206-211) ignores return
value by IERC20(openx).transfer(bar,amountOut.div(3)) (contracts/OpenXmaker.sol#210)
UniswapV2Router02.removeLiquidity(address,address,uint256,uint256,uint256,address,uint25
6) (contracts/uniswapv2/UniswapV2Router02.sol#105-121) ignores return value by
IUniswapV2Pair(pair).transferFrom(msg.sender,pair,liquidity) (contracts/uniswapv2/
UniswapV2Router02.sol#115)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-
transfer

TokenPaymentManager.getPaymentAmount(uint256) (contracts/
TreasuryFundManager.sol#261-263) performs a multiplication on the result of a division:
        -
initialFundingAmount.div(totalTime).mul(_timestamp.sub(lastDistributionTimestamp))
(contracts/TreasuryFundManager.sol#262)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply

OpxChef.pendingReward(uint256,address) (contracts/opx/opxChef.sol#462-473) performs a
multiplication on the result of a division:
        - sushiReward = elapsedTime.mul(REWARDS_PER_SEC).mul(pool.allocPoint) /
totalAllocPoint (contracts/opx/opxChef.sol#469)
        - accSushiPerShare = accSushiPerShare.add(sushiReward.mul(ACC_OPX_PRECISION) /
lpSupply) (contracts/opx/opxChef.sol#470)
OpxChef.updatePool(uint256) (contracts/opx/opxChef.sol#492-505) performs a
multiplication on the result of a division:
        - sushiReward = elapsedTime.mul(REWARDS_PER_SEC).mul(pool.allocPoint) /
totalAllocPoint (contracts/opx/opxChef.sol#498)
        - pool.accSushiPerShare =
pool.accSushiPerShare.add((sushiReward.mul(ACC_OPX_PRECISION) / lpSupply).to128())
```

```
(contracts/opx/opxChef.sol#499)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply

MasterChefV2O.pendingReward(uint256,address) (contracts/MasterChefV2O.sol#489-500)
performs a multiplication on the result of a division:
        - sushiReward = elapsedTime.mul(REWARDS_PER_SEC).mul(pool.allocPoint) /
totalAllocPoint (contracts/MasterChefV2O.sol#496)
        - accSushiPerShare =
accSushiPerShare.add(sushiReward.mul(ACC_OPENX_PRECISION) / lpSupply) (contracts/
MasterChefV2O.sol#497)
MasterChefV2O.updatePool(uint256) (contracts/MasterChefV2O.sol#519-532) performs a
multiplication on the result of a division:
        - sushiReward = elapsedTime.mul(REWARDS_PER_SEC).mul(pool.allocPoint) /
totalAllocPoint (contracts/MasterChefV2O.sol#525)
        - pool.accSushiPerShare =
pool.accSushiPerShare.add((sushiReward.mul(ACC_OPENX_PRECISION) / lpSupply).to128())
(contracts/MasterChefV2O.sol#526)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply

OpenXMaker._convertStep(address,address,uint256,uint256) (contracts/
OpenXmaker.sol#128-176) performs a multiplication on the result of a division:
        - IERC2OBURNABLE(openx).burn(amount0.div(3).mul(2)) (contracts/
OpenXmaker.sol#144)
OpenXMaker._convertStep(address,address,uint256,uint256) (contracts/
OpenXmaker.sol#128-176) performs a multiplication on the result of a division:
        - IERC2OBURNABLE(openx).burn(amount1.div(3).mul(2)) (contracts/
OpenXmaker.sol#149)
OpenXMaker._convertStep(address,address,uint256,uint256) (contracts/
OpenXmaker.sol#128-176) performs a multiplication on the result of a division:
        - IERC2OBURNABLE(openx).burn(amount.div(3).mul(2)) (contracts/
OpenXmaker.sol#133)
OpenXMaker._toOPENX(address,uint256) (contracts/OpenXmaker.sol#206-211) performs a
multiplication on the result of a division:
        - IERC2OBURNABLE(openx).burn(amountOut.div(3).mul(2)) (contracts/
OpenXmaker.sol#209)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply

OpenXGov.enter(uint256) (contracts/OpenXbar.sol#55-73) uses a dangerous strict
equality:
```

```
        - totalShares == 0 || totalOpenX == 0 (contracts/OpenXbar.sol#63)
UniswapV2Pair._safeTransfer(address,address,uint256) (contracts/uniswapv2/
UniswapV2Pair.sol#50-53) uses a dangerous strict equality:
        - require(bool,string)(success && (data.length == 0 || abi.decode(data,
(bool))),OpenSwapV2 TRANSFER_FAILED) (contracts/uniswapv2/UniswapV2Pair.sol#52)
UniswapV2Pair.mint(address) (contracts/uniswapv2/UniswapV2Pair.sol#123-144) uses a
dangerous strict equality:
        - _totalSupply == 0 (contracts/uniswapv2/UniswapV2Pair.sol#132)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-
strict-equalities

Reentrancy in OpxChef.compoundRewards(address) (contracts/opx/opxChef.sol#438-455):
        External calls:
        - _deposit(0,fee,owner) (contracts/opx/opxChef.sol#450)
                - _rewarder.onSushiReward(pid,to,to,0,user.amount) (contracts/opx/
opxChef.sol#543)
        - _deposit(0,_pendingOpx.sub(fee),_user) (contracts/opx/opxChef.sol#451)
                - _rewarder.onSushiReward(pid,to,to,0,user.amount) (contracts/opx/
opxChef.sol#543)
        State variables written after the call(s):
        - _deposit(0,_pendingOpx.sub(fee),_user) (contracts/opx/opxChef.sol#451)
                - poolInfo[pid] = pool (contracts/opx/opxChef.sol#502)
        - _deposit(0,_pendingOpx.sub(fee),_user) (contracts/opx/opxChef.sol#451)
                - totalDeposited = totalDeposited.add(amount) (contracts/opx/
opxChef.sol#536)
        - _deposit(0,_pendingOpx.sub(fee),_user) (contracts/opx/opxChef.sol#451)
                - user.amount = user.amount.add(amount) (contracts/opx/opxChef.sol#537)
                - user.rewardDebt =
user.rewardDebt.add(int256(amount.mul(pool.accSushiPerShare) / ACC_OPX_PRECISION))
(contracts/opx/opxChef.sol#538)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-1

Reentrancy in UniswapV2Pair.burn(address) (contracts/uniswapv2/
UniswapV2Pair.sol#147-169):
        External calls:
        - _safeTransfer(_token0,to,amount0) (contracts/uniswapv2/UniswapV2Pair.sol#161)
                - (success,data) =
token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/uniswapv2/
UniswapV2Pair.sol#51)
        - _safeTransfer(_token1,to,amount1) (contracts/uniswapv2/UniswapV2Pair.sol#162)
```

```
                - (success,data) =
token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/uniswapv2/
UniswapV2Pair.sol#51)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/uniswapv2/
UniswapV2Pair.sol#166)
                - blockTimestampLast = blockTimestamp (contracts/uniswapv2/
UniswapV2Pair.sol#97)
        - kLast = uint256(reserve0).mul(reserve1) (contracts/uniswapv2/
UniswapV2Pair.sol#167)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/uniswapv2/
UniswapV2Pair.sol#166)
                - reserve0 = uint112(balance0) (contracts/uniswapv2/
UniswapV2Pair.sol#95)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/uniswapv2/
UniswapV2Pair.sol#166)
                - reserve1 = uint112(balance1) (contracts/uniswapv2/
UniswapV2Pair.sol#96)
Reentrancy in UniswapV2Factory.createPair(address,address) (contracts/uniswapv2/
UniswapV2Factory.sol#29-45):
        External calls:
        - UniswapV2Pair(pair).initialize(token0,token1) (contracts/uniswapv2/
UniswapV2Factory.sol#40)
        State variables written after the call(s):
        - getPair[token0][token1] = pair (contracts/uniswapv2/UniswapV2Factory.sol#41)
        - getPair[token1][token0] = pair (contracts/uniswapv2/UniswapV2Factory.sol#42)
Reentrancy in OpenXGov.init(IMC) (contracts/OpenXbar.sol#79-91):
        External calls:
        - OpenX.transferFrom(msg.sender,address(this),285889 * 10 ** 18) (contracts/
OpenXbar.sol#83)
        State variables written after the call(s):
        - masterchef = _masterchef (contracts/OpenXbar.sol#85)
Reentrancy in UniswapV2Pair.swap(uint256,uint256,address,bytes) (contracts/uniswapv2/
UniswapV2Pair.sol#172-200):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (contracts/uniswapv2/
UniswapV2Pair.sol#183)
                - (success,data) =
token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/uniswapv2/
UniswapV2Pair.sol#51)
        - _safeTransfer(_token1,to,amount1Out) (contracts/uniswapv2/
UniswapV2Pair.sol#184)
```

```
                - (success,data) =
token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/uniswapv2/
UniswapV2Pair.sol#51)
        - IUniswapV2Callee(to).uniswapV2Call(msg.sender,amount0Out,amount1Out,data)
(contracts/uniswapv2/UniswapV2Pair.sol#185)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/uniswapv2/
UniswapV2Pair.sol#198)
                - blockTimestampLast = blockTimestamp (contracts/uniswapv2/
UniswapV2Pair.sol#97)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/uniswapv2/
UniswapV2Pair.sol#198)
                - reserve0 = uint112(balance0) (contracts/uniswapv2/
UniswapV2Pair.sol#95)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/uniswapv2/
UniswapV2Pair.sol#198)
                - reserve1 = uint112(balance1) (contracts/uniswapv2/
UniswapV2Pair.sol#96)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-1

UniswapV2Library.getAmountsOut(address,uint256,address[]).i (contracts/uniswapv2/
libraries/UniswapV2Library.sol#68) is a local variable never initialized
UniswapV2Router02._swap(uint256[],address[],address).i (contracts/uniswapv2/
UniswapV2Router02.sol#215) is a local variable never initialized
UniswapV2Router02._swapSupportingFeeOnTransferTokens(address[],address).i (contracts/
uniswapv2/UniswapV2Router02.sol#324) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-
local-variables

OpenXGov.init(IMC) (contracts/OpenXbar.sol#79-91) ignores return value by
dummyToken.approve(address(masterchef),balance) (contracts/OpenXbar.sol#89)
UniswapV2Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256)
(contracts/uniswapv2/UniswapV2Router02.sol#35-62) ignores return value by
IUniswapV2Factory(factory).createPair(tokenA,tokenB) (contracts/uniswapv2/
UniswapV2Router02.sol#45)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

ContractDeployer.deployCtrct(bytes32,bytes) (contracts/deployer.sol#12-20) uses
assembly
        - INLINE ASM (contracts/deployer.sol#14-16)
```

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

BaseBoringBatchable._getRevertMsg(bytes) (contracts/MasterChefV2O.sol#306-315) uses
assembly
        - INLINE ASM (contracts/MasterChefV2O.sol#310-313)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

UniswapV2ERC20.constructor() (contracts/uniswapv2/UniswapV2ERC20.sol#25-39) uses
assembly
        - INLINE ASM (contracts/uniswapv2/UniswapV2ERC20.sol#27-29)
UniswapV2Factory.createPair(address,address) (contracts/uniswapv2/
UniswapV2Factory.sol#29-45) uses assembly
        - INLINE ASM (contracts/uniswapv2/UniswapV2Factory.sol#36-38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

SafeMath.add(uint256,uint256) (contracts/TreasuryFundManager.sol#16-21) is never used
and should be removed
SafeMath.mod(uint256,uint256) (contracts/TreasuryFundManager.sol#126-128) is never used
and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/TreasuryFundManager.sol#142-145) is
never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

BoringERC20.returnDataToString(bytes) (contracts/opx/opxChef.sol#230-246) is never used
and should be removed
BoringERC20.safeDecimals(IERC20) (contracts/opx/opxChef.sol#267-270) is never used and
should be removed
BoringERC20.safeName(IERC20) (contracts/opx/opxChef.sol#259-262) is never used and
should be removed
BoringERC20.safeSymbol(IERC20) (contracts/opx/opxChef.sol#251-254) is never used and
should be removed
BoringMath.to32(uint256) (contracts/opx/opxChef.sol#124-127) is never used and should
be removed
BoringMath128.sub(uint128,uint128) (contracts/opx/opxChef.sol#136-138) is never used
and should be removed
SignedSafeMath.div(int256,int256) (contracts/opx/opxChef.sol#50-57) is never used and
should be removed
SignedSafeMath.mul(int256,int256) (contracts/opx/opxChef.sol#22-36) is never used and
should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
TransferHelper.safeApprove(address,address,uint256) (contracts/uniswapv2/libraries/
TransferHelper.sol#7-11) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version>=0.5.0 (contracts/uniswapv2/interfaces/IERC20.sol#3) allows old versions
Pragma version>=0.5.0 (contracts/uniswapv2/interfaces/IUniswapV2Callee.sol#3) allows
old versions
Pragma version>=0.5.0 (contracts/uniswapv2/interfaces/IUniswapV2ERC20.sol#3) allows old
versions
Pragma version>=0.5.0 (contracts/uniswapv2/interfaces/IUniswapV2Factory.sol#3) allows
old versions
Pragma version>=0.5.0 (contracts/uniswapv2/interfaces/IUniswapV2Pair.sol#3) allows old
versions
Pragma version>=0.6.2 (contracts/uniswapv2/interfaces/IUniswapV2Router01.sol#3) allows
old versions
Pragma version>=0.6.2 (contracts/uniswapv2/interfaces/IUniswapV2Router02.sol#3) allows
old versions
Pragma version>=0.5.0 (contracts/uniswapv2/interfaces/IWETH.sol#3) allows old versions
Pragma version>=0.6.0 (contracts/uniswapv2/libraries/TransferHelper.sol#3) allows old
versions
Pragma version>=0.5.0 (contracts/uniswapv2/libraries/UniswapV2Library.sol#3) allows old
versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity

Low level call in BoringERC20.safeSymbol(IERC20) (contracts/opx/opxChef.sol#251-254):
        - (success,data) =
address(token).staticcall(abi.encodeWithSelector(SIG_SYMBOL)) (contracts/opx/
opxChef.sol#252)
Low level call in BoringERC20.safeName(IERC20) (contracts/opx/opxChef.sol#259-262):
        - (success,data) = address(token).staticcall(abi.encodeWithSelector(SIG_NAME))
(contracts/opx/opxChef.sol#260)
Low level call in BoringERC20.safeDecimals(IERC20) (contracts/opx/opxChef.sol#267-270):
        - (success,data) =
address(token).staticcall(abi.encodeWithSelector(SIG_DECIMALS)) (contracts/opx/
opxChef.sol#268)
Low level call in BoringERC20.safeTransfer(IERC20,address,uint256) (contracts/opx/
opxChef.sol#277-284):
        - (success,data) =
address(token).call(abi.encodeWithSelector(SIG_TRANSFER,to,amount)) (contracts/opx/
opxChef.sol#282)
```

```
Low level call in BoringERC20.safeTransferFrom(IERC20,address,address,uint256)
(contracts/opx/opxChef.sol#292-300):
        - (success,data) =
address(token).call(abi.encodeWithSelector(SIG_TRANSFER_FROM,from,to,amount))
(contracts/opx/opxChef.sol#298)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls

Low level call in BoringERC20.safeSymbol(IERC20) (contracts/MasterChefV20.sol#251-254):
        - (success,data) =
address(token).staticcall(abi.encodeWithSelector(SIG_SYMBOL)) (contracts/
MasterChefV20.sol#252)
Low level call in BoringERC20.safeName(IERC20) (contracts/MasterChefV20.sol#259-262):
        - (success,data) = address(token).staticcall(abi.encodeWithSelector(SIG_NAME))
(contracts/MasterChefV20.sol#260)
Low level call in BoringERC20.safeDecimals(IERC20) (contracts/
MasterChefV20.sol#267-270):
        - (success,data) =
address(token).staticcall(abi.encodeWithSelector(SIG_DECIMALS)) (contracts/
MasterChefV20.sol#268)
Low level call in BoringERC20.safeTransfer(IERC20,address,uint256) (contracts/
MasterChefV20.sol#277-284):
        - (success,data) =
address(token).call(abi.encodeWithSelector(SIG_TRANSFER,to,amount)) (contracts/
MasterChefV20.sol#282)
Low level call in BoringERC20.safeTransferFrom(IERC20,address,address,uint256)
(contracts/MasterChefV20.sol#292-300):
        - (success,data) =
address(token).call(abi.encodeWithSelector(SIG_TRANSFER_FROM,from,to,amount))
(contracts/MasterChefV20.sol#298)
Low level call in BaseBoringBatchable.batch(bytes[],bool) (contracts/
MasterChefV20.sol#326-335):
        - (success,result) = address(this).delegatecall(calls[i]) (contracts/
MasterChefV20.sol#330)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls

Low level call in UniswapV2Pair._safeTransfer(address,address,uint256) (contracts/
uniswapv2/UniswapV2Pair.sol#50-53):
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value))
(contracts/uniswapv2/UniswapV2Pair.sol#51)
```

```
Low level call in TransferHelper.safeApprove(address,address,uint256) (contracts/
uniswapv2/libraries/TransferHelper.sol#7-11):
        - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value))
(contracts/uniswapv2/libraries/TransferHelper.sol#9)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (contracts/
uniswapv2/libraries/TransferHelper.sol#13-17):
        - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value))
(contracts/uniswapv2/libraries/TransferHelper.sol#15)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256)
(contracts/uniswapv2/libraries/TransferHelper.sol#19-23):
        - (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value))
(contracts/uniswapv2/libraries/TransferHelper.sol#21)
Low level call in TransferHelper.safeTransferETH(address,uint256) (contracts/uniswapv2/
libraries/TransferHelper.sol#25-28):
        - (success) = to.call{value: value}(new bytes(0)) (contracts/uniswapv2/
libraries/TransferHelper.sol#26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls

OpenX (contracts/OpenXToken.sol#5-16) should inherit from IERC20BURNABLE (contracts/
OpenXmaker.sol#12-14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-
inheritance

MasterChefV2O.slitherConstructorConstantVariables() (contracts/
MasterChefV2O.sol#367-658) uses literals with too many digits:
        - REWARDS_PER_SEC = 31709791980000000 (contracts/MasterChefV2O.sol#409)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-
digits

OpenX.constructor() (contracts/OpenXToken.sol#6-9) uses literals with too many digits:
        - _mint(msg.sender,166249999900000000000000000) (contracts/OpenXToken.sol#8)
UniswapV2Factory.pairCodeHash() (contracts/uniswapv2/UniswapV2Factory.sol#25-27) uses
literals with too many digits:
        - keccak256(bytes)(type()(UniswapV2Pair).creationCode) (contracts/uniswapv2/
UniswapV2Factory.sol#26)
UniswapV2Factory.createPair(address,address) (contracts/uniswapv2/
UniswapV2Factory.sol#29-45) uses literals with too many digits:
        - bytecode = type()(UniswapV2Pair).creationCode (contracts/uniswapv2/
UniswapV2Factory.sol#34)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-
digits
```

```
TokenPaymentManager.totalTime (contracts/TreasuryFundManager.sol#247) should be
constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-constant
```

0x Guard