



Smart contracts security assessment

Final report

[Tariff: Standard](#)

XcelSwap

May 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	10
8. Disclaimer	11
9. Slither output	12

Introduction

The report has been prepared for XcelSwap. Two contracts were tested: Timelock and XcelSwapMasterChef. XcelSwapMasterChef is a fork of the famous contracts of the same name from the PancakeSwap and SushiSwap projects. The code is available in the Github [repository](#). The code was checked in the [0354002](#) commit. Used libraries are realized with the use of OpenZeppelin libraries, which is considered the best practice.

Name	XcelSwap
Audit date	2022-05-19 - 2022-05-22
Language	Solidity
Platform	Binance Smart Chain

Contracts checked

Name	Address
Timelock	https://github.com/XcelSwap/xcelswap-contract-audit/blob/03540028fc215977929f6a1608e93df459eb11d1/contracts/Timelock.sol
XcelSwapMasterChef	https://github.com/XcelSwap/xcelswap-contract-audit/blob/03540028fc215977929f6a1608e93df459eb11d1/contracts/XcelSwapMasterChef.sol

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
<u>Unencrypted Private Data On-Chain</u>	passed
<u>Code With No Effects</u>	passed
<u>Message call with hardcoded gas amount</u>	passed
<u>Typographical Error</u>	passed
<u>DoS With Block Gas Limit</u>	passed
<u>Presence of unused variables</u>	passed
<u>Incorrect Inheritance Order</u>	passed
<u>Requirement Violation</u>	passed
<u>Weak Sources of Randomness from Chain Attributes</u>	passed
<u>Shadowing State Variables</u>	passed
<u>Incorrect Constructor Name</u>	passed
<u>Block values as a proxy for time</u>	passed
<u>Authorization through tx.origin</u>	passed
<u>DoS with Failed Call</u>	passed
<u>Delegatecall to Untrusted Callee</u>	passed

<u>Use of Deprecated Solidity Functions</u>	passed
<u>Assert Violation</u>	passed
<u>State Variable Default Visibility</u>	passed
<u>Reentrancy</u>	not passed
<u>Unprotected SELFDESTRUCT Instruction</u>	passed
<u>Unprotected Ether Withdrawal</u>	passed
<u>Unchecked Call Return Value</u>	passed
<u>Floating Pragma</u>	not passed
<u>Outdated Compiler Version</u>	passed
<u>Integer Overflow and Underflow</u>	passed
<u>Function Default Visibility</u>	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. Owner account compromised (XcelSwapMasterChef)

The main vulnerability here is that the `migrator` variable can be set to any contract without restrictions. The migrator can be installed by the owner using the `setMigrator()` function at any time and an unlimited number of times, which in the worst case (when the owner's account is compromised) can lead to the newly installed `migrator` being able to transfer all the base LP tokens to an arbitrary address.

Recommendation: The first good way to solve this problem would be to add a multi-person multi-signature to call this function, [AccessControl](#) from the OpenZeppelin library can help with this task. You also need to temporarily delay the function call so that the user has time to make appropriate decisions.

The second workaround is to completely remove the migration functionality since you can't be completely sure that the migrator's contract will be reassuring.

Low severity issues

1. Small minimum delay (Timelock)

The `MINIMUM_DELAY` constant has a time period of 6 hours which is too short. It is recommended to set the time to a day or more. The user should have more time to check for pending function calls.

2. No check for null address (Timelock)

To avoid accidental errors when using a contract or deploying it, it is recommended to check the address input parameters for zero values. 1) The constructor does not check the `admin_` parameter

for null address

2) The `setPendingAdmin()` function does not check the `pendingAdmin_` parameter for null address

Recommendation: It is recommended to add a check of input parameters to zero address using `require`.

3. Missing emit event (Timelock)

The constructor assigns the initial address to the `admin` variable, but does not fire the `NewAdmin()` event.

Recommendation: It is recommended to add a `NewAdmin()` event call to the function to log the initial address of the admin.

4. Adding identical pools (XcelSwapMasterChef)

If the token is mistakenly added more than once in the `add()` function, then the calculation of the expected reward in the `updatePool()` function will be incorrect, since the funds may be distributed to different pools.

5. Optional update (XcelSwapMasterChef)

If the `add()` or `set()` function is called with a negative `_withUpdate`, the calculation of rewards in other pools in the `updatePool()` function does not work correctly. It is recommended to remove the `_withUpdate` parameter in these functions and leave only the `massUpdatePools()` call.

6. Reentrancy (XcelSwapMasterChef)

Common case liquidity tokens do not have callback functions, technically reentrancy is possible when using functions utilizing the `safeTransfer()` method.

Recommendation: Consider importing OpenZeppelin ReentrancyGuard contract and adding a `nonReentrant` modifier to `deposit()`, `withdraw()`, `emergencyWithdraw()`,

`enterStaking()`, `leaveStaking()`.

7. Commission tokens are not supported (XcelSwapMasterChef)

If you try to enter tokens with a commission, then `user.amount` is incorrectly calculated in the `deposit()` function, which leads to the `withdraw()` function being able to withdraw more tokens than we have.

```
function deposit(uint256 _pid, uint256 _amount) public {
    ...
    if (_amount > 0) {
        pool.lptoken.safeTransferFrom(address(msg.sender), address(this), _amount);
        user.amount = user.amount.add(_amount); // !!! error
    }
    ...
}
```

Recommendation: It is recommended to add a balance calculation to the pool before and after calling `pool.lptoken.safeTransferFrom()` in order to subsequently calculate the correct amount without a fee.

```
function deposit(uint256 _pid, uint256 _amount) public {
    ...
    if (_amount > 0) {
        balanceBefore = pool.lptoken.balanceOf(address(this));
        pool.lptoken.safeTransferFrom(address(msg.sender), address(this), _amount);
        balanceAfter = pool.lptoken.balanceOf(address(this));
        user.amount = user.amount.add(balanceAfter.sub(balanceBefore));
    }
    ...
}
```

8. The number of deposited LP tokens is incorrectly calculated (XcelSwapMasterChef)

In the `updatePool()` function, the balance of LP tokens of this contract is requested in L1441, but if you transfer LP tokens directly to the address of the contract, the balance would be combined with

these tokens. This leads to the incorrect further calculation of the formulas.

Recommendation: It is recommended to solve this problem by adding a new uint256 field to the **PoolInfo** structure, in which the number of coins entered by the user will be added on each deposit and subtracted when withdrawing.

```
struct PoolInfo {  
    IERC20 lpToken;  
    uint256 allocPoint;  
    uint256 lastRewardBlock;  
    uint256 accSushiPerShare;  
    uint256 lpTotalSupply; // new field  
}
```

9. Possible lack of gas (XcelSwapMasterChef)

The **massUpdatePools()** function cycles through the update of each pool from the PoolInfo array. If there are too many pools, then there may not be enough gas to process this function, which causes the pools to stop updating and correctly calculate the staking reward.

10. No check for null address (XcelSwapMasterChef)

To avoid accidental errors when using a contract or deploying it, it is recommended to check the address input parameters for zero values. 1) The constructor does not check the **__devaddr** parameter for null address

2) **dev()** function does not check the **__devaddr** parameter for null address

Recommendation: It is recommended to add a check of input parameters to zero address using **require**.

Conclusion

XcelSwap Timelock, XcelSwapMasterChef contracts were audited. 1 medium, 10 low severity issues were found.

We strongly suggest adding unit and functional tests for all contracts.

We also recommend using pragma fixed to the version the contracts have been tested and are intended to be deployed with. This helps to avoid deploying using an outdated compiler version and shields from possible bugs in future solidity releases.

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

```
MasterChef.safeXldTransfer(address,uint256) (contracts/
XcelSwapMasterChef.sol#1551-1560) ignores return value by xld.transfer(_to,xldBal)
(contracts/XcelSwapMasterChef.sol#1554)
MasterChef.safeXldTransfer(address,uint256) (contracts/
XcelSwapMasterChef.sol#1551-1560) ignores return value by xld.transfer(_to,_amount)
(contracts/XcelSwapMasterChef.sol#1556)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
```

```
MasterChef.pendingCake(uint256,address) (contracts/XcelSwapMasterChef.sol#1415-1426)
performs a multiplication on the result of a division:
    -xldReward =
multiplier.mul(xldPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (contracts/
XcelSwapMasterChef.sol#1422)
    -accCakePerShare = accCakePerShare.add(xldReward.mul(1e12).div(lpSupply))
(contracts/XcelSwapMasterChef.sol#1423)
MasterChef.updatePool(uint256) (contracts/XcelSwapMasterChef.sol#1438-1454) performs a
multiplication on the result of a division:
    -xldReward =
multiplier.mul(xldPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (contracts/
XcelSwapMasterChef.sol#1449)
    -pool.accCakePerShare =
pool.accCakePerShare.add(xldReward.mul(1e12).div(lpSupply)) (contracts/
XcelSwapMasterChef.sol#1452)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
```

```
XcelDefiToken._writeCheckpoint(address,uint32,uint256,uint256) (contracts/
XcelSwapMasterChef.sol#1195-1213) uses a dangerous strict equality:
    - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock ==
blockNumber (contracts/XcelSwapMasterChef.sol#1205)
MasterChef.migrate(uint256) (contracts/XcelSwapMasterChef.sol#1398-1407) uses a
dangerous strict equality:
    - require(bool,string)(bal == newLpToken.balanceOf(address(this)),migrate: bad)
(contracts/XcelSwapMasterChef.sol#1405)
MasterChef.updatePool(uint256) (contracts/XcelSwapMasterChef.sol#1438-1454) uses a
dangerous strict equality:
```

```
- lpSupply == 0 (contracts/XcelSwapMasterChef.sol#1444)
```

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

Reentrancy in MasterChef.add(uint256,IBEP20,bool) (contracts/XcelSwapMasterChef.sol#1351-1364):

External calls:

```
- massUpdatePools() (contracts/XcelSwapMasterChef.sol#1353)
  - xld.mint(devaddr,xldReward.div(10)) (contracts/
```

```
XcelSwapMasterChef.sol#1450)
```

```
- xld.mint(address(this),xldReward) (contracts/
```

```
XcelSwapMasterChef.sol#1451)
```

State variables written after the call(s):

```
- poolInfo.push(PoolInfo(_lpToken,_allocPoint,lastRewardBlock,0)) (contracts/
```

```
XcelSwapMasterChef.sol#1357-1362)
```

```
- updateStakingPool() (contracts/XcelSwapMasterChef.sol#1363)
```

```
- poolInfo[0].allocPoint = points (contracts/
```

```
XcelSwapMasterChef.sol#1388)
```

```
- totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/
```

```
XcelSwapMasterChef.sol#1356)
```

```
- updateStakingPool() (contracts/XcelSwapMasterChef.sol#1363)
```

```
- totalAllocPoint =
```

```
totalAllocPoint.sub(poolInfo[0].allocPoint).add(points) (contracts/
```

```
XcelSwapMasterChef.sol#1387)
```

Reentrancy in MasterChef.deposit(uint256,uint256) (contracts/

XcelSwapMasterChef.sol#1457-1476):

External calls:

```
- updatePool(_pid) (contracts/XcelSwapMasterChef.sol#1463)
```

```
- xld.mint(devaddr,xldReward.div(10)) (contracts/
```

```
XcelSwapMasterChef.sol#1450)
```

```
- xld.mint(address(this),xldReward) (contracts/
```

```
XcelSwapMasterChef.sol#1451)
```

```
- safeXldTransfer(msg.sender,pending) (contracts/XcelSwapMasterChef.sol#1467)
```

```
- xld.transfer(_to,xldBal) (contracts/XcelSwapMasterChef.sol#1554)
```

```
- xld.transfer(_to,_amount) (contracts/XcelSwapMasterChef.sol#1556)
```

```
- pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
```

```
(contracts/XcelSwapMasterChef.sol#1471)
```

State variables written after the call(s):

```
- user.amount = user.amount.add(_amount) (contracts/
```

```
XcelSwapMasterChef.sol#1472)
```

```
- user.rewardDebt = user.amount.mul(pool.accCakePerShare).div(1e12) (contracts/
```

```
XcelSwapMasterChef.sol#1474)
```

Reentrancy in MasterChef.emergencyWithdraw(uint256) (contracts/
XcelSwapMasterChef.sol#1541-1548):

External calls:

- pool.lpToken.safeTransfer(address(msg.sender),user.amount) (contracts/
XcelSwapMasterChef.sol#1544)

State variables written after the call(s):

- user.amount = 0 (contracts/XcelSwapMasterChef.sol#1546)
- user.rewardDebt = 0 (contracts/XcelSwapMasterChef.sol#1547)

Reentrancy in MasterChef.enterStaking(uint256) (contracts/
XcelSwapMasterChef.sol#1500-1518):

External calls:

- updatePool(0) (contracts/XcelSwapMasterChef.sol#1503)
- xld.mint(devaddr,xldReward.div(10)) (contracts/
XcelSwapMasterChef.sol#1450)
- xld.mint(address(this),xldReward) (contracts/
XcelSwapMasterChef.sol#1451)

- safeXldTransfer(msg.sender,pending) (contracts/XcelSwapMasterChef.sol#1507)
- xld.transfer(_to,xldBal) (contracts/XcelSwapMasterChef.sol#1554)
- xld.transfer(_to,_amount) (contracts/XcelSwapMasterChef.sol#1556)
- pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)

(contracts/XcelSwapMasterChef.sol#1511)

State variables written after the call(s):

- user.amount = user.amount.add(_amount) (contracts/
XcelSwapMasterChef.sol#1512)
- user.rewardDebt = user.amount.mul(pool.accCakePerShare).div(1e12) (contracts/
XcelSwapMasterChef.sol#1514)

Reentrancy in MasterChef.leaveStaking(uint256) (contracts/
XcelSwapMasterChef.sol#1521-1538):

External calls:

- updatePool(0) (contracts/XcelSwapMasterChef.sol#1525)
- xld.mint(devaddr,xldReward.div(10)) (contracts/
XcelSwapMasterChef.sol#1450)
- xld.mint(address(this),xldReward) (contracts/
XcelSwapMasterChef.sol#1451)

- safeXldTransfer(msg.sender,pending) (contracts/XcelSwapMasterChef.sol#1528)
- xld.transfer(_to,xldBal) (contracts/XcelSwapMasterChef.sol#1554)
- xld.transfer(_to,_amount) (contracts/XcelSwapMasterChef.sol#1556)

State variables written after the call(s):

- user.amount = user.amount.sub(_amount) (contracts/
XcelSwapMasterChef.sol#1531)

Reentrancy in MasterChef.leaveStaking(uint256) (contracts/
XcelSwapMasterChef.sol#1521-1538):

External calls:

```
- updatePool(0) (contracts/XcelSwapMasterChef.sol#1525)
  - xld.mint(devaddr,xldReward.div(10)) (contracts/
XcelSwapMasterChef.sol#1450)
  - xld.mint(address(this),xldReward) (contracts/
XcelSwapMasterChef.sol#1451)
  - safeXldTransfer(msg.sender,pending) (contracts/XcelSwapMasterChef.sol#1528)
  - xld.transfer(_to,xldBal) (contracts/XcelSwapMasterChef.sol#1554)
  - xld.transfer(_to,_amount) (contracts/XcelSwapMasterChef.sol#1556)
  - pool.lpToken.safeTransfer(address(msg.sender),_amount) (contracts/
XcelSwapMasterChef.sol#1532)
```

State variables written after the call(s):

```
- user.rewardDebt = user.amount.mul(pool.accCakePerShare).div(1e12) (contracts/
XcelSwapMasterChef.sol#1534)
```

Reentrancy in MasterChef.migrate(uint256) (contracts/XcelSwapMasterChef.sol#1398-1407):

External calls:

```
- lpToken.safeApprove(address(migrator),bal) (contracts/
XcelSwapMasterChef.sol#1403)
- newLpToken = migrator.migrate(lpToken) (contracts/
XcelSwapMasterChef.sol#1404)
```

State variables written after the call(s):

```
- pool.lpToken = newLpToken (contracts/XcelSwapMasterChef.sol#1406)
```

Reentrancy in MasterChef.set(uint256,uint256,bool) (contracts/
XcelSwapMasterChef.sol#1367-1377):

External calls:

```
- massUpdatePools() (contracts/XcelSwapMasterChef.sol#1369)
  - xld.mint(devaddr,xldReward.div(10)) (contracts/
XcelSwapMasterChef.sol#1450)
  - xld.mint(address(this),xldReward) (contracts/
XcelSwapMasterChef.sol#1451)
```

State variables written after the call(s):

```
- poolInfo[_pid].allocPoint = _allocPoint (contracts/
XcelSwapMasterChef.sol#1372)
- updateStakingPool() (contracts/XcelSwapMasterChef.sol#1375)
  - poolInfo[0].allocPoint = points (contracts/
XcelSwapMasterChef.sol#1388)
  - totalAllocPoint = totalAllocPoint.sub(prevAllocPoint).add(_allocPoint)
(contracts/XcelSwapMasterChef.sol#1374)
  - updateStakingPool() (contracts/XcelSwapMasterChef.sol#1375)
    - totalAllocPoint =
totalAllocPoint.sub(poolInfo[0].allocPoint).add(points) (contracts/
XcelSwapMasterChef.sol#1387)
```

```

Reentrancy in MasterChef.updatePool(uint256) (contracts/
XcelSwapMasterChef.sol#1438-1454):
    External calls:
    - xld.mint(devaddr,xldReward.div(10)) (contracts/XcelSwapMasterChef.sol#1450)
    - xld.mint(address(this),xldReward) (contracts/XcelSwapMasterChef.sol#1451)
    State variables written after the call(s):
    - pool.accCakePerShare =
pool.accCakePerShare.add(xldReward.mul(1e12).div(lpSupply)) (contracts/
XcelSwapMasterChef.sol#1452)
    - pool.lastRewardBlock = block.number (contracts/XcelSwapMasterChef.sol#1453)
Reentrancy in MasterChef.withdraw(uint256,uint256) (contracts/
XcelSwapMasterChef.sol#1479-1497):
    External calls:
    - updatePool(_pid) (contracts/XcelSwapMasterChef.sol#1486)
      - xld.mint(devaddr,xldReward.div(10)) (contracts/
XcelSwapMasterChef.sol#1450)
      - xld.mint(address(this),xldReward) (contracts/
XcelSwapMasterChef.sol#1451)
    - safeXldTransfer(msg.sender,pending) (contracts/XcelSwapMasterChef.sol#1489)
      - xld.transfer(_to,xldBal) (contracts/XcelSwapMasterChef.sol#1554)
      - xld.transfer(_to,_amount) (contracts/XcelSwapMasterChef.sol#1556)
    State variables written after the call(s):
    - user.amount = user.amount.sub(_amount) (contracts/
XcelSwapMasterChef.sol#1492)
Reentrancy in MasterChef.withdraw(uint256,uint256) (contracts/
XcelSwapMasterChef.sol#1479-1497):
    External calls:
    - updatePool(_pid) (contracts/XcelSwapMasterChef.sol#1486)
      - xld.mint(devaddr,xldReward.div(10)) (contracts/
XcelSwapMasterChef.sol#1450)
      - xld.mint(address(this),xldReward) (contracts/
XcelSwapMasterChef.sol#1451)
    - safeXldTransfer(msg.sender,pending) (contracts/XcelSwapMasterChef.sol#1489)
      - xld.transfer(_to,xldBal) (contracts/XcelSwapMasterChef.sol#1554)
      - xld.transfer(_to,_amount) (contracts/XcelSwapMasterChef.sol#1556)
    - pool.lpToken.safeTransfer(address(msg.sender),_amount) (contracts/
XcelSwapMasterChef.sol#1493)
    State variables written after the call(s):
    - user.rewardDebt = user.amount.mul(pool.accCakePerShare).div(1e12) (contracts/
XcelSwapMasterChef.sol#1495)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-1

```


BEP20.constructor(string,string).name (contracts/XcelSwapMasterChef.sol#739) shadows:

- BEP20.name() (contracts/XcelSwapMasterChef.sol#755-757) (function)
- IBEP20.name() (contracts/XcelSwapMasterChef.sol#27) (function)

BEP20.constructor(string,string).symbol (contracts/XcelSwapMasterChef.sol#739) shadows:

- BEP20.symbol() (contracts/XcelSwapMasterChef.sol#763-765) (function)
- IBEP20.symbol() (contracts/XcelSwapMasterChef.sol#22) (function)

BEP20.allowance(address,address).owner (contracts/XcelSwapMasterChef.sol#804) shadows:

- Ownable.owner() (contracts/XcelSwapMasterChef.sol#642-644) (function)

BEP20._approve(address,address,uint256).owner (contracts/XcelSwapMasterChef.sol#963)

shadows:

- Ownable.owner() (contracts/XcelSwapMasterChef.sol#642-644) (function)

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

MasterChef.updateMultiplier(uint256) (contracts/XcelSwapMasterChef.sol#1341-1343)

should emit an event for:

- BONUS_MULTIPLIER = multiplierNumber (contracts/XcelSwapMasterChef.sol#1342)

MasterChef.add(uint256,IBEP20,bool) (contracts/XcelSwapMasterChef.sol#1351-1364) should emit an event for:

- totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/

XcelSwapMasterChef.sol#1356)

MasterChef.set(uint256,uint256,bool) (contracts/XcelSwapMasterChef.sol#1367-1377)

should emit an event for:

- totalAllocPoint = totalAllocPoint.sub(prevAllocPoint).add(_allocPoint)

(contracts/XcelSwapMasterChef.sol#1374)

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

Timelock.constructor(address,uint256).admin_ (contracts/Timelock.sol#235) lacks a zero-check on :

- admin = admin_ (contracts/Timelock.sol#239)

Timelock.setPendingAdmin(address).pendingAdmin_ (contracts/Timelock.sol#264) lacks a zero-check on :

- pendingAdmin = pendingAdmin_ (contracts/Timelock.sol#272)

Timelock.executeTransaction(address,uint256,string,bytes,uint256).target (contracts/Timelock.sol#297) lacks a zero-check on :

- (success,returnData) = target.call.value(value)(callData) (contracts/

Timelock.sol#316)

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

MasterChef.constructor(XcelDefiToken,address,uint256,uint256)._devaddr (contracts/XcelSwapMasterChef.sol#1319) lacks a zero-check on :

- devaddr = _devaddr (contracts/XcelSwapMasterChef.sol#1325)

MasterChef.dev(address)._devaddr (contracts/XcelSwapMasterChef.sol#1563) lacks a zero-check on :

- devaddr = _devaddr (contracts/XcelSwapMasterChef.sol#1565)

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

MasterChef.updatePool(uint256) (contracts/XcelSwapMasterChef.sol#1438-1454) has external calls inside a loop: lpSupply = pool.lpToken.balanceOf(address(this)) (contracts/XcelSwapMasterChef.sol#1443)

MasterChef.updatePool(uint256) (contracts/XcelSwapMasterChef.sol#1438-1454) has external calls inside a loop: xld.mint(devaddr,xldReward.div(10)) (contracts/XcelSwapMasterChef.sol#1450)

MasterChef.updatePool(uint256) (contracts/XcelSwapMasterChef.sol#1438-1454) has external calls inside a loop: xld.mint(address(this),xldReward) (contracts/XcelSwapMasterChef.sol#1451)

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

Reentrancy in Timelock.executeTransaction(address,uint256,string,bytes,uint256) (contracts/Timelock.sol#297-322):

External calls:

- (success,returnData) = target.call.value(value)(callData) (contracts/Timelock.sol#316)

Event emitted after the call(s):

- ExecuteTransaction(txHash,target,value,signature,data,eta) (contracts/Timelock.sol#319)

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

Reentrancy in MasterChef.deposit(uint256,uint256) (contracts/XcelSwapMasterChef.sol#1457-1476):

External calls:

- updatePool(_pid) (contracts/XcelSwapMasterChef.sol#1463)
 - xld.mint(devaddr,xldReward.div(10)) (contracts/XcelSwapMasterChef.sol#1450)
 - xld.mint(address(this),xldReward) (contracts/XcelSwapMasterChef.sol#1451)

```

- safeXldTransfer(msg.sender,pending) (contracts/XcelSwapMasterChef.sol#1467)
  - xld.transfer(_to,xldBal) (contracts/XcelSwapMasterChef.sol#1554)
  - xld.transfer(_to,_amount) (contracts/XcelSwapMasterChef.sol#1556)
- pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
(contracts/XcelSwapMasterChef.sol#1471)
Event emitted after the call(s):
- Deposit(msg.sender,_pid,_amount) (contracts/XcelSwapMasterChef.sol#1475)
Reentrancy in MasterChef.emergencyWithdraw(uint256) (contracts/
XcelSwapMasterChef.sol#1541-1548):
External calls:
- pool.lpToken.safeTransfer(address(msg.sender),user.amount) (contracts/
XcelSwapMasterChef.sol#1544)
Event emitted after the call(s):
- EmergencyWithdraw(msg.sender,_pid,user.amount) (contracts/
XcelSwapMasterChef.sol#1545)
Reentrancy in MasterChef.enterStaking(uint256) (contracts/
XcelSwapMasterChef.sol#1500-1518):
External calls:
- updatePool(0) (contracts/XcelSwapMasterChef.sol#1503)
  - xld.mint(devaddr,xldReward.div(10)) (contracts/
XcelSwapMasterChef.sol#1450)
  - xld.mint(address(this),xldReward) (contracts/
XcelSwapMasterChef.sol#1451)
- safeXldTransfer(msg.sender,pending) (contracts/XcelSwapMasterChef.sol#1507)
  - xld.transfer(_to,xldBal) (contracts/XcelSwapMasterChef.sol#1554)
  - xld.transfer(_to,_amount) (contracts/XcelSwapMasterChef.sol#1556)
- pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
(contracts/XcelSwapMasterChef.sol#1511)
Event emitted after the call(s):
- Deposit(msg.sender,0,_amount) (contracts/XcelSwapMasterChef.sol#1517)
Reentrancy in MasterChef.leaveStaking(uint256) (contracts/
XcelSwapMasterChef.sol#1521-1538):
External calls:
- updatePool(0) (contracts/XcelSwapMasterChef.sol#1525)
  - xld.mint(devaddr,xldReward.div(10)) (contracts/
XcelSwapMasterChef.sol#1450)
  - xld.mint(address(this),xldReward) (contracts/
XcelSwapMasterChef.sol#1451)
- safeXldTransfer(msg.sender,pending) (contracts/XcelSwapMasterChef.sol#1528)
  - xld.transfer(_to,xldBal) (contracts/XcelSwapMasterChef.sol#1554)
  - xld.transfer(_to,_amount) (contracts/XcelSwapMasterChef.sol#1556)

```

```

- pool.lptoken.safeTransfer(address(msg.sender),_amount) (contracts/
XcelSwapMasterChef.sol#1532)
  Event emitted after the call(s):
- Withdraw(msg.sender,0,_amount) (contracts/XcelSwapMasterChef.sol#1537)
Reentrancy in MasterChef.withdraw(uint256,uint256) (contracts/
XcelSwapMasterChef.sol#1479-1497):
  External calls:
- updatePool(_pid) (contracts/XcelSwapMasterChef.sol#1486)
  - xld.mint(devaddr,xldReward.div(10)) (contracts/
XcelSwapMasterChef.sol#1450)
  - xld.mint(address(this),xldReward) (contracts/
XcelSwapMasterChef.sol#1451)
- safeXldTransfer(msg.sender,pending) (contracts/XcelSwapMasterChef.sol#1489)
  - xld.transfer(_to,xldBal) (contracts/XcelSwapMasterChef.sol#1554)
  - xld.transfer(_to,_amount) (contracts/XcelSwapMasterChef.sol#1556)
- pool.lptoken.safeTransfer(address(msg.sender),_amount) (contracts/
XcelSwapMasterChef.sol#1493)
  Event emitted after the call(s):
- Withdraw(msg.sender,_pid,_amount) (contracts/XcelSwapMasterChef.sol#1496)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3

```

```

Timelock.queueTransaction(address,uint256,string,bytes,uint256) (contracts/
Timelock.sol#277-286) uses timestamp for comparisons

```

Dangerous comparisons:

```

- require(bool,string)(eta >=
getBlockTimestamp().add(delay),Timelock::queueTransaction: Estimated execution block
must satisfy delay.) (contracts/Timelock.sol#279)

```

```

Timelock.executeTransaction(address,uint256,string,bytes,uint256) (contracts/
Timelock.sol#297-322) uses timestamp for comparisons

```

Dangerous comparisons:

```

- require(bool,string)(getBlockTimestamp() >= eta,Timelock::executeTransaction:
Transaction hasn't surpassed time lock.) (contracts/Timelock.sol#302)
- require(bool,string)(getBlockTimestamp() <=
eta.add(GRACE_PERIOD),Timelock::executeTransaction: Transaction is stale.) (contracts/
Timelock.sol#303)

```

```

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

```

```

XcelDefiToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (contracts/
XcelSwapMasterChef.sol#1061-1102) uses timestamp for comparisons

```

Dangerous comparisons:

- require(bool,string)(now <= expiry,XLD::delegateBySig: signature expired) (contracts/XcelSwapMasterChef.sol#1100)

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

Address.isContract(address) (contracts/XcelSwapMasterChef.sol#341-350) uses assembly

- INLINE ASM (contracts/XcelSwapMasterChef.sol#348)

Address._verifyCallResult(bool,bytes,string) (contracts/XcelSwapMasterChef.sol#486-503) uses assembly

- INLINE ASM (contracts/XcelSwapMasterChef.sol#495-498)

XcelDefiToken.getChainId() (contracts/XcelSwapMasterChef.sol#1220-1224) uses assembly

- INLINE ASM (contracts/XcelSwapMasterChef.sol#1222)

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

Different versions of Solidity is used:

- Version used: ['0.6.12', '>=0.4.0']
- >=0.4.0 (contracts/TimeLock.sol#7)
- 0.6.12 (contracts/TimeLock.sol#208)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used>

Different versions of Solidity is used:

- Version used: ['0.6.12', '>=0.4.0', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0', '>=0.6.4']
- >=0.6.4 (contracts/XcelSwapMasterChef.sol#6)
- >=0.6.0<0.8.0 (contracts/XcelSwapMasterChef.sol#102)
- >=0.6.2<0.8.0 (contracts/XcelSwapMasterChef.sol#318)
- >=0.6.0<0.8.0 (contracts/XcelSwapMasterChef.sol#509)
- >=0.6.0<0.8.0 (contracts/XcelSwapMasterChef.sol#585)
- >=0.6.0<0.8.0 (contracts/XcelSwapMasterChef.sol#611)
- >=0.6.0<0.8.0 (contracts/XcelSwapMasterChef.sol#680)
- >=0.4.0 (contracts/XcelSwapMasterChef.sol#685)
- 0.6.12 (contracts/XcelSwapMasterChef.sol#985)
- 0.6.12 (contracts/XcelSwapMasterChef.sol#1229)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used>

SafeMath.div(uint256,uint256) (contracts/TimeLock.sol#111-113) is never used and should be removed

SafeMath.div(uint256,uint256,string) (contracts/TimeLock.sol#127-137) is never used and

should be removed

SafeMath.min(uint256,uint256) (contracts/TimeLock.sol#176-178) is never used and should be removed

SafeMath.mod(uint256,uint256) (contracts/TimeLock.sol#151-153) is never used and should be removed

SafeMath.mod(uint256,uint256,string) (contracts/TimeLock.sol#167-174) is never used and should be removed

SafeMath.mul(uint256,uint256) (contracts/TimeLock.sol#85-97) is never used and should be removed

SafeMath.sqrt(uint256) (contracts/TimeLock.sol#181-192) is never used and should be removed

SafeMath.sub(uint256,uint256) (contracts/TimeLock.sol#50-52) is never used and should be removed

SafeMath.sub(uint256,uint256,string) (contracts/TimeLock.sol#64-73) is never used and should be removed

Reference: <https://github.com/crytic/sliether/wiki/Detector-Documentation#dead-code>

Address.functionCall(address,bytes) (contracts/XcelSwapMasterChef.sol#394-396) is never used and should be removed

Address.functionCallWithValue(address,bytes,uint256) (contracts/XcelSwapMasterChef.sol#419-421) is never used and should be removed

Address.functionDelegateCall(address,bytes) (contracts/XcelSwapMasterChef.sol#468-470) is never used and should be removed

Address.functionDelegateCall(address,bytes,string) (contracts/XcelSwapMasterChef.sol#478-484) is never used and should be removed

Address.functionStaticCall(address,bytes) (contracts/XcelSwapMasterChef.sol#444-446) is never used and should be removed

Address.functionStaticCall(address,bytes,string) (contracts/XcelSwapMasterChef.sol#454-460) is never used and should be removed

Address.sendValue(address,uint256) (contracts/XcelSwapMasterChef.sol#368-374) is never used and should be removed

BEP20._burn(address,uint256) (contracts/XcelSwapMasterChef.sol#942-948) is never used and should be removed

BEP20._burnFrom(address,uint256) (contracts/XcelSwapMasterChef.sol#977-980) is never used and should be removed

Context._msgData() (contracts/XcelSwapMasterChef.sol#602-605) is never used and should be removed

SafeBEP20.safeDecreaseAllowance(IBEP20,address,uint256) (contracts/XcelSwapMasterChef.sol#558-561) is never used and should be removed

SafeBEP20.safeIncreaseAllowance(IBEP20,address,uint256) (contracts/XcelSwapMasterChef.sol#553-556) is never used and should be removed

SafeMath.tryAdd(uint256,uint256) (contracts/XcelSwapMasterChef.sol#123-127) is never used and should be removed

SafeMath.tryDiv(uint256,uint256) (contracts/XcelSwapMasterChef.sol#159-162) is never used and should be removed

SafeMath.tryMod(uint256,uint256) (contracts/XcelSwapMasterChef.sol#169-172) is never used and should be removed

SafeMath.tryMul(uint256,uint256) (contracts/XcelSwapMasterChef.sol#144-152) is never used and should be removed

SafeMath.trySub(uint256,uint256) (contracts/XcelSwapMasterChef.sol#134-137) is never used and should be removed

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

Pragma version>=0.4.0 (contracts/Timelock.sol#7) allows old versions

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity>

Pragma version>=0.6.4 (contracts/XcelSwapMasterChef.sol#6) allows old versions

Pragma version>=0.6.0<0.8.0 (contracts/XcelSwapMasterChef.sol#102) is too complex

Pragma version>=0.6.2<0.8.0 (contracts/XcelSwapMasterChef.sol#318) is too complex

Pragma version>=0.6.0<0.8.0 (contracts/XcelSwapMasterChef.sol#509) is too complex

Pragma version>=0.6.0<0.8.0 (contracts/XcelSwapMasterChef.sol#585) is too complex

Pragma version>=0.6.0<0.8.0 (contracts/XcelSwapMasterChef.sol#611) is too complex

Pragma version>=0.6.0<0.8.0 (contracts/XcelSwapMasterChef.sol#680) is too complex

Pragma version>=0.4.0 (contracts/XcelSwapMasterChef.sol#685) allows old versions

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity>

Low level call in Timelock.executeTransaction(address,uint256,string,bytes,uint256) (contracts/Timelock.sol#297-322):

- (success,returnData) = target.call.value(value)(callData) (contracts/Timelock.sol#316)

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

Low level call in Address.sendValue(address,uint256) (contracts/XcelSwapMasterChef.sol#368-374):

- (success) = recipient.call{value: amount}() (contracts/XcelSwapMasterChef.sol#372)

Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (contracts/XcelSwapMasterChef.sol#429-436):

- (success,returndata) = target.call{value: value}(data) (contracts/XcelSwapMasterChef.sol#434)

Low level call in Address.functionStaticCall(address,bytes,string) (contracts/XcelSwapMasterChef.sol#454-460):

- (success, returndata) = target.staticcall(data) (contracts/XcelSwapMasterChef.sol#458)

Low level call in Address.functionDelegateCall(address,bytes,string) (contracts/XcelSwapMasterChef.sol#478-484):

- (success, returndata) = target.delegatecall(data) (contracts/XcelSwapMasterChef.sol#482)

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

Variable Timelock.admin_initialized (contracts/Timelock.sol#230) is not in mixedCase

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

Parameter XcelDefiToken.mint(address,uint256)._to (contracts/XcelSwapMasterChef.sol#991) is not in mixedCase

Parameter XcelDefiToken.mint(address,uint256)._amount (contracts/XcelSwapMasterChef.sol#991) is not in mixedCase

Variable XcelDefiToken._delegates (contracts/XcelSwapMasterChef.sol#1003) is not in mixedCase

Parameter MasterChef.add(uint256,IBEP20,bool)._allocPoint (contracts/XcelSwapMasterChef.sol#1351) is not in mixedCase

Parameter MasterChef.add(uint256,IBEP20,bool)._lpToken (contracts/XcelSwapMasterChef.sol#1351) is not in mixedCase

Parameter MasterChef.add(uint256,IBEP20,bool)._withUpdate (contracts/XcelSwapMasterChef.sol#1351) is not in mixedCase

Parameter MasterChef.set(uint256,uint256,bool)._pid (contracts/XcelSwapMasterChef.sol#1367) is not in mixedCase

Parameter MasterChef.set(uint256,uint256,bool)._allocPoint (contracts/XcelSwapMasterChef.sol#1367) is not in mixedCase

Parameter MasterChef.set(uint256,uint256,bool)._withUpdate (contracts/XcelSwapMasterChef.sol#1367) is not in mixedCase

Parameter MasterChef.setMigrator(IMigratorChef)._migrator (contracts/XcelSwapMasterChef.sol#1393) is not in mixedCase

Parameter MasterChef.migrate(uint256)._pid (contracts/XcelSwapMasterChef.sol#1398) is not in mixedCase

Parameter MasterChef.getMultiplier(uint256,uint256)._from (contracts/XcelSwapMasterChef.sol#1410) is not in mixedCase

Parameter MasterChef.getMultiplier(uint256,uint256)._to (contracts/XcelSwapMasterChef.sol#1410) is not in mixedCase

Parameter MasterChef.pendingCake(uint256,address)._pid (contracts/XcelSwapMasterChef.sol#1415) is not in mixedCase

Parameter MasterChef.pendingCake(uint256,address)._user (contracts/XcelSwapMasterChef.sol#1415) is not in mixedCase

Parameter MasterChef.updatePool(uint256)._pid (contracts/XcelSwapMasterChef.sol#1438) is not in mixedCase

Parameter MasterChef.deposit(uint256,uint256)._pid (contracts/XcelSwapMasterChef.sol#1457) is not in mixedCase

Parameter MasterChef.deposit(uint256,uint256)._amount (contracts/XcelSwapMasterChef.sol#1457) is not in mixedCase

Parameter MasterChef.withdraw(uint256,uint256)._pid (contracts/XcelSwapMasterChef.sol#1479) is not in mixedCase

Parameter MasterChef.withdraw(uint256,uint256)._amount (contracts/XcelSwapMasterChef.sol#1479) is not in mixedCase

Parameter MasterChef.enterStaking(uint256)._amount (contracts/XcelSwapMasterChef.sol#1500) is not in mixedCase

Parameter MasterChef.leaveStaking(uint256)._amount (contracts/XcelSwapMasterChef.sol#1521) is not in mixedCase

Parameter MasterChef.emergencyWithdraw(uint256)._pid (contracts/XcelSwapMasterChef.sol#1541) is not in mixedCase

Parameter MasterChef.safeXldTransfer(address,uint256)._to (contracts/XcelSwapMasterChef.sol#1551) is not in mixedCase

Parameter MasterChef.safeXldTransfer(address,uint256)._amount (contracts/XcelSwapMasterChef.sol#1551) is not in mixedCase

Parameter MasterChef.dev(address)._devaddr (contracts/XcelSwapMasterChef.sol#1563) is not in mixedCase

Variable MasterChef.BONUS_MULTIPLIER (contracts/XcelSwapMasterChef.sol#1299) is not in mixedCase

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

Redundant expression "this (contracts/XcelSwapMasterChef.sol#603)" inContext (contracts/XcelSwapMasterChef.sol#597-606)

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

setDelay(uint256) should be declared external:

- Timelock.setDelay(uint256) (contracts/Timelock.sol#247-254)

acceptAdmin() should be declared external:

- Timelock.acceptAdmin() (contracts/Timelock.sol#256-262)

setPendingAdmin(address) should be declared external:

```

- Timelock.setPendingAdmin(address) (contracts/Timelock.sol#264-275)
queueTransaction(address,uint256,string,bytes,uint256) should be declared external:
- Timelock.queueTransaction(address,uint256,string,bytes,uint256) (contracts/
Timelock.sol#277-286)
cancelTransaction(address,uint256,string,bytes,uint256) should be declared external:
- Timelock.cancelTransaction(address,uint256,string,bytes,uint256) (contracts/
Timelock.sol#288-295)
executeTransaction(address,uint256,string,bytes,uint256) should be declared external:
- Timelock.executeTransaction(address,uint256,string,bytes,uint256) (contracts/
Timelock.sol#297-322)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

renounceOwnership() should be declared external:
- Ownable.renounceOwnership() (contracts/XcelSwapMasterChef.sol#661-664)
transferOwnership(address) should be declared external:
- Ownable.transferOwnership(address) (contracts/XcelSwapMasterChef.sol#670-674)
symbol() should be declared external:
- BEP20.symbol() (contracts/XcelSwapMasterChef.sol#763-765)
decimals() should be declared external:
- BEP20.decimals() (contracts/XcelSwapMasterChef.sol#770-772)
totalSupply() should be declared external:
- BEP20.totalSupply() (contracts/XcelSwapMasterChef.sol#777-779)
transfer(address,uint256) should be declared external:
- BEP20.transfer(address,uint256) (contracts/XcelSwapMasterChef.sol#796-799)
allowance(address,address) should be declared external:
- BEP20.allowance(address,address) (contracts/XcelSwapMasterChef.sol#804-806)
approve(address,uint256) should be declared external:
- BEP20.approve(address,uint256) (contracts/XcelSwapMasterChef.sol#815-818)
transferFrom(address,address,uint256) should be declared external:
- BEP20.transferFrom(address,address,uint256) (contracts/
XcelSwapMasterChef.sol#832-840)
increaseAllowance(address,uint256) should be declared external:
- BEP20.increaseAllowance(address,uint256) (contracts/
XcelSwapMasterChef.sol#854-857)
decreaseAllowance(address,uint256) should be declared external:
- BEP20.decreaseAllowance(address,uint256) (contracts/
XcelSwapMasterChef.sol#873-876)
mint(uint256) should be declared external:
- BEP20.mint(uint256) (contracts/XcelSwapMasterChef.sol#886-889)
mint(address,uint256) should be declared external:

```

```

- XcelDefiToken.mint(address,uint256) (contracts/XcelSwapMasterChef.sol#991-994)
updateMultiplier(uint256) should be declared external:
- MasterChef.updateMultiplier(uint256) (contracts/
XcelSwapMasterChef.sol#1341-1343)
add(uint256,IBEP20,bool) should be declared external:
- MasterChef.add(uint256,IBEP20,bool) (contracts/
XcelSwapMasterChef.sol#1351-1364)
set(uint256,uint256,bool) should be declared external:
- MasterChef.set(uint256,uint256,bool) (contracts/
XcelSwapMasterChef.sol#1367-1377)
setMigrator(IMigratorChef) should be declared external:
- MasterChef.setMigrator(IMigratorChef) (contracts/
XcelSwapMasterChef.sol#1393-1395)
migrate(uint256) should be declared external:
- MasterChef.migrate(uint256) (contracts/XcelSwapMasterChef.sol#1398-1407)
deposit(uint256,uint256) should be declared external:
- MasterChef.deposit(uint256,uint256) (contracts/
XcelSwapMasterChef.sol#1457-1476)
withdraw(uint256,uint256) should be declared external:
- MasterChef.withdraw(uint256,uint256) (contracts/
XcelSwapMasterChef.sol#1479-1497)
enterStaking(uint256) should be declared external:
- MasterChef.enterStaking(uint256) (contracts/XcelSwapMasterChef.sol#1500-1518)
leaveStaking(uint256) should be declared external:
- MasterChef.leaveStaking(uint256) (contracts/XcelSwapMasterChef.sol#1521-1538)
emergencyWithdraw(uint256) should be declared external:
- MasterChef.emergencyWithdraw(uint256) (contracts/
XcelSwapMasterChef.sol#1541-1548)
dev(address) should be declared external:
- MasterChef.dev(address) (contracts/XcelSwapMasterChef.sol#1563-1566)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

```



 Guard