0x Guard

# Smart contracts security assessment

Final report

Tariff: Standard

## Champion Finance EVIC

September 2022

0xguard.com

hello@0xguard.com

## Contents

## Introduction

The report has been prepared for Champion Finance.

The Champion Finance Protocol allows users to farm  EVICTokens. The EVICToken is a rebase tokens. The EVICToken owner (taxOffice) can set fee on token trading.

Contracts EvicTreasury and EvicBoardroom allow keeping a stable price of the EVICToken using rebase mechanism.

The code is available at the GitHub repository and was audited after the commit 9a66fa1228782f3453473e5fa26148722657a48e.

**Report Update.**

The contracts code was updated according to this report and rechecked after the commit 9c35b7ac0cf1848cdd1a45cc1efac16b84c7ce9a.

Only 4 contracts with its dependencies were audited: EVICToken, EVICGenesis, EVICBoardroomV2, EVICTreasuryV2.sol.

| Name | Champion Finance EVIC |
|---|---|
| Audit date | 2022-08-25 - 2022-09-06 |
| Language | Solidity |
| Platform | Avalanche Network |

## Contracts checked

| Name | Address |
|---|---|
| EVICToken | 0x74FeFa839A96A1632A29E0fcf0907d0F88528658 |
| EVICGenesis | 0x26dDE1A20944e9D067a3DCeF60fd23673C246671 |

| | |
|---|---|
| EVICBoardroomV2 | 0x6001Ca31953459704ba7eA44A9387f68B4f1B639, 0x98EBb6cEd9db54b11EEc7cc7136fA07743D118ef |
| EVICTreasuryV2 | 0x543230e268A95838d1F8abC2aC1F2E986F871631 |

## 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 | passed |
| Reentrancy | passed |
| Unprotected SELFDESTRUCT Instruction | passed |
| Unprotected Ether Withdrawal | passed |
| Unchecked Call Return Value | passed |
| Floating Pragma | passed |
| Outdated Compiler Version | passed |
| Integer Overflow and Underflow | passed |
| Function Default Visibility | passed |

## Classification of issue severity

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

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

## Issues

### High severity issues

**1. Blacklist (EVICToken)**
Status: Fixed

The contract owner (`taxOffice`) has the ability to block any user by adding him to blacklist ( `addToBlackList()`). This is not fair, since users who have invested in the project may be blocked simply at the request of the owner.

Also this functionality can be used for rugpull.

**Recommendation:** It is necessary to restrict the owner's rights to use the blacklist functionality.

**2. Changing oracle (EVICToken)**
Status: Fixed

The contract operator can change the implementation of the oracle using the `setTokenOracle()` function. This can lead to a completely different calculation of the weighted token price.

```
function setTokenOracle(address _oracle) external onlyOperator {
    oracle = _oracle;
}
```

**Recommendation:** Restrict the operator's (owner) ability to change the oracle contract address.

### 3. Unlimited tax parameters (EVICToken)

Status: Fixed

The contract admin (`taxOffice`) is able to set `taxRateAfterRebase` for 100% (or even more) and to set a huge value of `timeTaxAfterRebase`.

**Recommendation:** Add validation for parameters of the `setTaxRateAfterRebase()`, `setTimeTaxAfterRebase()` functions.

## Medium severity issues

### 1. Fee checking (EVICToken)

Status: Fixed

The tax fee amount should be checked in the `setTaxTiersRate()` function instead of being checked during token transfers in the `calculateTaxRate()` function (L425). Because users will not be able to transfer tokens due to an admin mistake.

**Recommendation:** We recommend replacing `taxTiersRates` check into `setTaxTiersRate()` function.

## Low severity issues

### 1. Redundant check of the uint variable (EVICToken)

Status: Fixed

There are checks for `uint` value in L435 and L450.

```
require(_index >= 0, "Index has to be higher than 0");
```

This check does nothing because `uint` value >= 0 by default.

**Recommendation:** Check the logic of the functions, and make sure these checks are necessary. Perhaps there was supposed to be a different functionality.

**2. Gas optimization (EVICToken)**
Status: Open

The vilibility of the `setMarketLpPairs()` function can be changed to external to save gas.

**3. Non-validated values (EVICBoardroomV2)**
Status: Fixed

The `setWithdrawLockupEpoch()` and `setRewardLockupEpoch()` functions have been added to the updated code.

Consider adding validation for the input parameters of these functions. Because some users may not have time to use the claim functions.

**4. Payouts to funds over 100% (EVICTreasuryV2)**
Status: Open

The contract operator can set 99% reward amount for each fund: `daoFundSharedPercent` and `polFundSharedPercent`.

Also, together their sum can exceed 100% (up to 198%). Thus, the function `_sendToBoardroom()` will always fail and block part of the project, because fund payments will exceed `_amount` (L337).

```
function _sendToBoardroom(uint256 _amount) internal {
    IEVICToken mainTokenErc20 = IEVICToken(mainToken);
    mainTokenErc20.mint(address(this), _amount);

    uint256 _daoFundSharedAmount = _amount.mul(daoFundSharedPercent).div(100);
    address daoFund = mainTokenErc20.getDaoFund();
    mainTokenErc20.transfer(daoFund, _daoFundSharedAmount);
    emit DaoFundFunded(block.timestamp, _daoFundSharedAmount);

    uint256 _polFundSharedAmount = _amount.mul(polFundSharedPercent).div(100);
    address polFund = mainTokenErc20.getPolWallet();
    mainTokenErc20.transfer(polFund, _polFundSharedAmount);
    emit PolFundFunded(block.timestamp, _polFundSharedAmount);

    _amount = _amount.sub(_daoFundSharedAmount).sub(_polFundSharedAmount);
    ...
    }
```

**Recommendation:** It is necessary to decrease the validation threshold (current value is 100) of the `setDaoFundSharedPercent()`, `setPolFundSharedPercent()` functions.

```
require(_value < 100, 'Treasury: Max percent is 100%');
```

**Update:** In the updated code the contract operator can set 50% reward amount for each fund: `daoFundSharedPercent` and `polFundSharedPercent` (in total 100%). This means that all awards will go to these funds. And there will be nothing left for the all boardroom contracts.

Consider the possibility of lowering the total payment to the funds.

The severity of the issue has been lowered in the update.

## Conclusion

Champion Finance EVIC EVICToken, EVICGenesis, EVICBoardroomV2, EVICTreasuryV2 contracts were audited. 3 high, 1 medium, 4 low severity issues were found.
3 high, 1 medium, 2 low severity issues have been fixed in the update.

According to this report 3 high, 1 medium and 2 low issues were fixed by the Champion Finance team.

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.

## Slither's output

```
EVICTreasuryV2._sendToBoardroom(uint256) (contracts/EVICTreasuryV2.sol#323-361) ignores
return value by mainTokenErc20.transfer(daoFund,_daoFundSharedAmount) (contracts/
EVICTreasuryV2.sol#329)
EVICTreasuryV2._sendToBoardroom(uint256) (contracts/EVICTreasuryV2.sol#323-361) ignores
return value by mainTokenErc20.transfer(polFund,_polFundSharedAmount) (contracts/
EVICTreasuryV2.sol#334)
EVICTreasuryV2._sendToBoardroom(uint256) (contracts/EVICTreasuryV2.sol#323-361) ignores
return value by mainTokenErc20.transfer(daoFund,daoFundReward) (contracts/
EVICTreasuryV2.sol#355)
EmergencyWithdraw.emergencyWithdrawTokenBalance(address,address,uint256) (contracts/
utils/EmergencyWithdraw.sol#45-52) ignores return value by erc20.transfer(_to,_amount)
(contracts/utils/EmergencyWithdraw.sol#51)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-
transfer

EVICTreasuryV2.getEstimatedReward(uint256) (contracts/EVICTreasuryV2.sol#299-321)
performs a multiplication on the result of a division:
        -estimatedReward = mainTokenTotalSupply.mul(percentage).div(10000) (contracts/
EVICTreasuryV2.sol#304)
        -estimatedReward = estimatedReward.mul(expansionRate).div(10000) (contracts/
EVICTreasuryV2.sol#308)
EVICTreasuryV2.getEstimatedReward(uint256) (contracts/EVICTreasuryV2.sol#299-321)
performs a multiplication on the result of a division:
        -estimatedReward = estimatedReward.mul(expansionRate).div(10000) (contracts/
EVICTreasuryV2.sol#308)
        -_daoFundSharedAmount = estimatedReward.mul(daoFundSharedPercent).div(100)
(contracts/EVICTreasuryV2.sol#314)
EVICTreasuryV2.getEstimatedReward(uint256) (contracts/EVICTreasuryV2.sol#299-321)
performs a multiplication on the result of a division:
        -estimatedReward = estimatedReward.mul(expansionRate).div(10000) (contracts/
EVICTreasuryV2.sol#308)
        -_polFundSharedAmount = estimatedReward.mul(polFundSharedPercent).div(100)
(contracts/EVICTreasuryV2.sol#315)
EVICTreasuryV2.getEstimatedReward(uint256) (contracts/EVICTreasuryV2.sol#299-321)
performs a multiplication on the result of a division:
        -estimatedReward = estimatedReward.mul(expansionRate).div(10000) (contracts/
EVICTreasuryV2.sol#308)
```

```
        -estimatedReward.mul(boardroomPool.allocPoint).div(totalAllocPoint) (contracts/
EVICTreasuryV2.sol#320)
EVICTreasuryV2.allocateSeigniorage() (contracts/EVICTreasuryV2.sol#398-449) performs a
multiplication on the result of a division:
        -_savedForBoardroom = mainTokenTotalSupply.mul(_percentage).div(10000)
(contracts/EVICTreasuryV2.sol#409)
        -_savedForBoardroom = _savedForBoardroom.mul(expansionRate).div(10000)
(contracts/EVICTreasuryV2.sol#414)
EVICTreasuryV2.computeSupplyDelta() (contracts/EVICTreasuryV2.sol#451-464) performs a
multiplication on the result of a division:
        -rebasePercentage = targetRate.sub(rate).mul(ONE).div(targetRate) (contracts/
EVICTreasuryV2.sol#458)
        -supplyDelta =
mathRound(getMainTokenCirculatingSupply().mul(rebasePercentage).div(ONE)) (contracts/
EVICTreasuryV2.sol#463)
EVICTreasuryV2.mathRound(uint256) (contracts/EVICTreasuryV2.sol#466-474) performs a
multiplication on the result of a division:
        -valueFloor = _value.div(midpointRounding).mul(midpointRounding) (contracts/
EVICTreasuryV2.sol#467)
EVICGenesis.pending(uint256,address) (contracts/distribution/EVICGenesis.sol#167-183)
performs a multiplication on the result of a division:
        -_evicTokenReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint)
(contracts/distribution/EVICGenesis.sol#174)
        -accEVICTokenPerShare =
accEVICTokenPerShare.add(_evicTokenReward.mul(1e18).div(tokenSupply)) (contracts/
distribution/EVICGenesis.sol#175)
EVICGenesis.updatePool(uint256) (contracts/distribution/EVICGenesis.sol#228-248)
performs a multiplication on the result of a division:
        -_evicTokenReward = _generatedReward.mul(pool.allocPoint).div(totalAllocPoint)
(contracts/distribution/EVICGenesis.sol#244)
        -pool.accEVICTokenPerShare =
pool.accEVICTokenPerShare.add(_evicTokenReward.mul(1e18).div(tokenSupply)) (contracts/
distribution/EVICGenesis.sol#245)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply

EVICGenesis.updatePool(uint256) (contracts/distribution/EVICGenesis.sol#228-248) uses a
dangerous strict equality:
        - tokenSupply == 0 (contracts/distribution/EVICGenesis.sol#234)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-
strict-equalities
```

```
Reentrancy in EVICTreasuryV2._rebase(uint256) (contracts/EVICTreasuryV2.sol#476-495):
        External calls:
        - newTotalSupply = IEVICToken(mainToken).rebase(epoch,supplyDelta,negative)
(contracts/EVICTreasuryV2.sol#488)
        State variables written after the call(s):
        - previousEpoch = epoch (contracts/EVICTreasuryV2.sol#490)
Reentrancy in EVICTreasuryV2.initialize(address,address,uint256) (contracts/
EVICTreasuryV2.sol#193-219):
        External calls:
        - IEVICToken(mainToken).grantRebaseExclusion(address(this)) (contracts/
EVICTreasuryV2.sol#213)
        State variables written after the call(s):
        - initialized = true (contracts/EVICTreasuryV2.sol#215)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-1

EVICToken.calculateTaxRate(uint256) (contracts/EVICToken.sol#440-454) contains a
tautology or contradiction:
        - tierId >= 0 (contracts/EVICToken.sol#446)
EVICTreasuryV2.calculateMaxSupplyExpansionPercent(uint256) (contracts/
EVICTreasuryV2.sol#363-380) contains a tautology or contradiction:
        - tierId >= 0 (contracts/EVICTreasuryV2.sol#372)
EVICTreasuryV2.calculateExpansionRate(uint256) (contracts/EVICTreasuryV2.sol#382-396)
contains a tautology or contradiction:
        - tierId >= 0 (contracts/EVICTreasuryV2.sol#388)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-
contradiction

EVICTreasuryV2.getTwapPrice().price (contracts/EVICTreasuryV2.sol#186) is a local
variable never initialized
EVICTreasuryV2.calculateExpansionRate(uint256).expansionRate (contracts/
EVICTreasuryV2.sol#383) is a local variable never initialized
EVICToken._getTokenPrice()._price (contracts/EVICToken.sol#413) is a local variable
never initialized
EVICTreasuryV2.calculateMaxSupplyExpansionPercent(uint256).maxSupplyExpansionPercent
(contracts/EVICTreasuryV2.sol#367) is a local variable never initialized
EVICTreasuryV2.getMainTokenPrice().price (contracts/EVICTreasuryV2.sol#178) is a local
variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-
local-variables
```

```
EVICToken._getTokenPrice() (contracts/EVICToken.sol#412-418) ignores return value by
IOracle(oracle).consult(address(this),1e18) (contracts/EVICToken.sol#413-417)
EVICTreasuryV2.getMainTokenPrice() (contracts/EVICTreasuryV2.sol#177-183) ignores
return value by IOracle(oracle).consult(mainToken,1e18) (contracts/
EVICTreasuryV2.sol#178-182)
EVICTreasuryV2.getTwapPrice() (contracts/EVICTreasuryV2.sol#185-191) ignores return
value by IOracle(oracle).twap(mainToken,1e18) (contracts/EVICTreasuryV2.sol#186-190)
EVICTreasuryV2._sendToBoardroom(uint256) (contracts/EVICTreasuryV2.sol#323-361) ignores
return value by mainTokenErc20.mint(address(this),_amount) (contracts/
EVICTreasuryV2.sol#325)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

EVICGenesis.add(uint256,IERC20,uint256) (contracts/distribution/
EVICGenesis.sol#112-139) should emit an event for:
        - totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/distribution/
EVICGenesis.sol#137)
EVICGenesis.set(uint256,uint256) (contracts/distribution/EVICGenesis.sol#142-150)
should emit an event for:
        - totalAllocPoint = totalAllocPoint.sub(pool.allocPoint).add(_allocPoint)
(contracts/distribution/EVICGenesis.sol#146)
EVICGenesis.setPoolStartTime(uint256) (contracts/distribution/EVICGenesis.sol#349-356)
should emit an event for:
        - poolStartTime = _poolStartTime (contracts/distribution/EVICGenesis.sol#353)
        - poolEndTime = poolStartTime + runningTime (contracts/distribution/
EVICGenesis.sol#354)
        - lastAirdropRewardTime = poolStartTime (contracts/distribution/
EVICGenesis.sol#355)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic

EVICBoardroomV2.setOperator(address)._operator (contracts/EVICBoardroomV2.sol#155)
lacks a zero-check on :
                - operator = _operator (contracts/EVICBoardroomV2.sol#156)
EVICTreasuryV2.setOperator(address)._operator (contracts/EVICTreasuryV2.sol#221) lacks
a zero-check on :
                - operator = _operator (contracts/EVICTreasuryV2.sol#222)
EmergencyWithdraw.emergencyWithdrawEthBalance(address,uint256)._to (contracts/utils/
EmergencyWithdraw.sol#28) lacks a zero-check on :
                - address(_to).transfer(_amount) (contracts/utils/
EmergencyWithdraw.sol#29)
```

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

EVICTreasuryV2.setWithdrawLockupEpoch(uint256) (contracts/EVICTreasuryV2.sol#539-544)
has external calls inside a loop:
IBoardroom(boardroomInfo[pid].boardroom).setWithdrawLockupEpoch(_value) (contracts/
EVICTreasuryV2.sol#542)
EVICTreasuryV2.setRewardLockupEpoch(uint256) (contracts/EVICTreasuryV2.sol#546-551) has
external calls inside a loop:
IBoardroom(boardroomInfo[pid].boardroom).setRewardLockupEpoch(_value) (contracts/
EVICTreasuryV2.sol#549)
EVICGenesis.updatePool(uint256) (contracts/distribution/EVICGenesis.sol#228-248) has
external calls inside a loop: tokenSupply = pool.token.balanceOf(address(this))
(contracts/distribution/EVICGenesis.sol#233)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-
a-loop

Variable 'EVICToken._getTokenPrice()._price (contracts/EVICToken.sol#413)' in
EVICToken._getTokenPrice() (contracts/EVICToken.sol#412-418) potentially used before
declaration: uint256(_price) (contracts/EVICToken.sol#414)
Variable 'EVICTreasuryV2.getMainTokenPrice().price (contracts/EVICTreasuryV2.sol#178)'
in EVICTreasuryV2.getMainTokenPrice() (contracts/EVICTreasuryV2.sol#177-183)
potentially used before declaration: uint256(price) (contracts/EVICTreasuryV2.sol#179)
Variable 'EVICTreasuryV2.getTwapPrice().price (contracts/EVICTreasuryV2.sol#186)' in
EVICTreasuryV2.getTwapPrice() (contracts/EVICTreasuryV2.sol#185-191) potentially used
before declaration: uint256(price) (contracts/EVICTreasuryV2.sol#187)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-
declaration-usage-of-local-variables

Reentrancy in EVICTreasuryV2._rebase(uint256) (contracts/EVICTreasuryV2.sol#476-495):
        External calls:
        - newTotalSupply = IEVICToken(mainToken).rebase(epoch,supplyDelta,negative)
(contracts/EVICTreasuryV2.sol#488)
        State variables written after the call(s):
        - epochRebases.push(epoch) (contracts/EVICTreasuryV2.sol#491)
Reentrancy in EVICToken._transfer(address,address,uint256) (contracts/
EVICToken.sol#327-359):
        External calls:
        - _updatePrice() (contracts/EVICToken.sol#343)
                - IOracle(oracle).update() (contracts/EVICToken.sol#421-423)
        State variables written after the call(s):
```

```
        - _transferBase(from,polWallet,taxAmount) (contracts/EVICToken.sol#354)
                - _balances[from] = _balances[from].sub(gonValue) (contracts/
EVICToken.sol#320)
                - _balances[to] = _balances[to].add(gonValue) (contracts/
EVICToken.sol#321)
        - _transferBase(from,to,amount) (contracts/EVICToken.sol#358)
                - _balances[from] = _balances[from].sub(gonValue) (contracts/
EVICToken.sol#320)
                - _balances[to] = _balances[to].add(gonValue) (contracts/
EVICToken.sol#321)
Reentrancy in EVICTreasuryV2.allocateSeigniorage() (contracts/
EVICTreasuryV2.sol#398-449):
        External calls:
        - _updatePrice() (contracts/EVICTreasuryV2.sol#399)
                - IOracle(oracle).update() (contracts/EVICTreasuryV2.sol#290-292)
        State variables written after the call(s):
        - previousEpochMainPrice = getMainTokenPrice() (contracts/
EVICTreasuryV2.sol#401)
        - totalEpochAbovePeg = totalEpochAbovePeg.add(1) (contracts/
EVICTreasuryV2.sol#403)
Reentrancy in EVICTreasuryV2.initialize(address,address,uint256) (contracts/
EVICTreasuryV2.sol#193-219):
        External calls:
        - IEVICToken(mainToken).grantRebaseExclusion(address(this)) (contracts/
EVICTreasuryV2.sol#213)
        State variables written after the call(s):
        - operator = msg.sender (contracts/EVICTreasuryV2.sol#216)
Reentrancy in EVICToken.transferFrom(address,address,uint256) (contracts/
EVICToken.sol#379-387):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/EVICToken.sol#384)
                - IOracle(oracle).update() (contracts/EVICToken.sol#421-423)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),allowance(sender,_msgSender()).sub(amount,ERC20:
transfer amount exceeds allowance)) (contracts/EVICToken.sol#385)
                - _allowances[owner][spender] = amount (contracts/EVICToken.sol#303)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-2

Reentrancy in EVICTreasuryV2._rebase(uint256) (contracts/EVICTreasuryV2.sol#476-495):
        External calls:
```

```
        - newTotalSupply = IEVICToken(mainToken).rebase(epoch,supplyDelta,negative)
(contracts/EVICTreasuryV2.sol#488)
        Event emitted after the call(s):
        - LogRebase(epoch,supplyDelta,targetRate,_oldPrice,newTotalSupply,oldTotalSupply
,block.timestamp) (contracts/EVICTreasuryV2.sol#494)
Reentrancy in EVICTreasuryV2._sendToBoardroom(uint256) (contracts/
EVICTreasuryV2.sol#323-361):
        External calls:
        - mainTokenErc20.mint(address(this),_amount) (contracts/EVICTreasuryV2.sol#325)
        - mainTokenErc20.transfer(daoFund,_daoFundSharedAmount) (contracts/
EVICTreasuryV2.sol#329)
        Event emitted after the call(s):
        - DaoFundFunded(block.timestamp,_daoFundSharedAmount) (contracts/
EVICTreasuryV2.sol#330)
Reentrancy in EVICTreasuryV2._sendToBoardroom(uint256) (contracts/
EVICTreasuryV2.sol#323-361):
        External calls:
        - mainTokenErc20.mint(address(this),_amount) (contracts/EVICTreasuryV2.sol#325)
        - mainTokenErc20.transfer(daoFund,_daoFundSharedAmount) (contracts/
EVICTreasuryV2.sol#329)
        - mainTokenErc20.transfer(polFund,_polFundSharedAmount) (contracts/
EVICTreasuryV2.sol#334)
        Event emitted after the call(s):
        - BoardroomFunded(block.timestamp,_amount) (contracts/EVICTreasuryV2.sol#360)
        - PolFundFunded(block.timestamp,_polFundSharedAmount) (contracts/
EVICTreasuryV2.sol#335)
Reentrancy in EVICToken._transfer(address,address,uint256) (contracts/
EVICToken.sol#327-359):
        External calls:
        - _updatePrice() (contracts/EVICToken.sol#343)
                - IOracle(oracle).update() (contracts/EVICToken.sol#421-423)
        Event emitted after the call(s):
        - Transfer(from,to,amount) (contracts/EVICToken.sol#322)
                - _transferBase(from,polWallet,taxAmount) (contracts/EVICToken.sol#354)
        - Transfer(from,to,amount) (contracts/EVICToken.sol#322)
                - _transferBase(from,to,amount) (contracts/EVICToken.sol#358)
Reentrancy in EVICTreasuryV2.addBoardroom(address,uint256) (contracts/
EVICTreasuryV2.sol#233-239):
        External calls:
        - IEVICToken(mainToken).grantRebaseExclusion(_boardroom) (contracts/
EVICTreasuryV2.sol#237)
```

```
        Event emitted after the call(s):
        - AddBoardroom(msg.sender,_boardroom,_allocPoint) (contracts/
EVICTreasuryV2.sol#238)
Reentrancy in EVICBoardroomV2.allocateSeigniorage(uint256) (contracts/
EVICBoardroomV2.sol#249-262):
        External calls:
        - rewardToken.safeTransferFrom(msg.sender,address(this),amount) (contracts/
EVICBoardroomV2.sol#260)
        Event emitted after the call(s):
        - RewardAdded(msg.sender,amount) (contracts/EVICBoardroomV2.sol#261)
Reentrancy in EVICBoardroomV2.claimReward() (contracts/EVICBoardroomV2.sol#238-247):
        External calls:
        - rewardToken.safeTransfer(msg.sender,reward) (contracts/
EVICBoardroomV2.sol#244)
        Event emitted after the call(s):
        - RewardPaid(msg.sender,reward) (contracts/EVICBoardroomV2.sol#245)
Reentrancy in EVICGenesis.emergencyWithdraw(uint256) (contracts/distribution/
EVICGenesis.sol#339-347):
        External calls:
        - pool.token.safeTransfer(msg.sender,_amount) (contracts/distribution/
EVICGenesis.sol#345)
        Event emitted after the call(s):
        - EmergencyWithdraw(msg.sender,_pid,_amount) (contracts/distribution/
EVICGenesis.sol#346)
Reentrancy in EVICTreasuryV2.initialize(address,address,uint256) (contracts/
EVICTreasuryV2.sol#193-219):
        External calls:
        - IEVICToken(mainToken).grantRebaseExclusion(address(this)) (contracts/
EVICTreasuryV2.sol#213)
        Event emitted after the call(s):
        - Initialized(msg.sender,block.number) (contracts/EVICTreasuryV2.sol#218)
Reentrancy in EVICToken.transferFrom(address,address,uint256) (contracts/
EVICToken.sol#379-387):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/EVICToken.sol#384)
                - IOracle(oracle).update() (contracts/EVICToken.sol#421-423)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/EVICToken.sol#304)
                -
_approve(sender,_msgSender(),allowance(sender,_msgSender()).sub(amount,ERC20: transfer
amount exceeds allowance)) (contracts/EVICToken.sol#385)
```

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

EVICToken.calculateTaxRate(uint256) (contracts/EVICToken.sol#440-454) uses timestamp
for comparisons
        Dangerous comparisons:
        - block.timestamp >= lastTimeRebase && block.timestamp <
lastTimeRebase.add(timeTaxAfterRebase) (contracts/EVICToken.sol#443)
EVICGenesis.constructor(address,address,uint256) (contracts/distribution/
EVICGenesis.sol#76-97) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp < _poolStartTime,late) (contracts/
distribution/EVICGenesis.sol#80)
EVICGenesis.checkPoolDuplicate(IERC20) (contracts/distribution/EVICGenesis.sol#104-109)
uses timestamp for comparisons
        Dangerous comparisons:
        - pid < length (contracts/distribution/EVICGenesis.sol#106)
        - require(bool,string)(poolInfo[pid].token != _token,GenesisPool: existing
pool?) (contracts/distribution/EVICGenesis.sol#107)
EVICGenesis.add(uint256,IERC20,uint256) (contracts/distribution/
EVICGenesis.sol#112-139) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp < poolStartTime (contracts/distribution/EVICGenesis.sol#119)
        - _lastRewardTime == 0 (contracts/distribution/EVICGenesis.sol#121)
        - _lastRewardTime < poolStartTime (contracts/distribution/EVICGenesis.sol#124)
        - _lastRewardTime == 0 || _lastRewardTime < block.timestamp (contracts/
distribution/EVICGenesis.sol#130)
        - _isStarted = (_lastRewardTime <= poolStartTime) || (_lastRewardTime <=
block.timestamp) (contracts/distribution/EVICGenesis.sol#134)
EVICGenesis.getGeneratedReward(uint256,uint256) (contracts/distribution/
EVICGenesis.sol#153-164) uses timestamp for comparisons
        Dangerous comparisons:
        - _fromTime >= _toTime (contracts/distribution/EVICGenesis.sol#154)
        - _toTime >= poolEndTime (contracts/distribution/EVICGenesis.sol#155)
        - _toTime <= poolStartTime (contracts/distribution/EVICGenesis.sol#160)
EVICGenesis.pending(uint256,address) (contracts/distribution/EVICGenesis.sol#167-183)
uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > pool.lastRewardTime && tokenSupply != 0 (contracts/
distribution/EVICGenesis.sol#172)
EVICGenesis.pendingAirdrop(uint256,uint256,address) (contracts/distribution/
EVICGenesis.sol#185-200) uses timestamp for comparisons
        Dangerous comparisons:
        - _fromTime >= _toTime (contracts/distribution/EVICGenesis.sol#187)
        - _toTime >= poolEndTime (contracts/distribution/EVICGenesis.sol#188)
        - _toTime <= poolStartTime (contracts/distribution/EVICGenesis.sol#193)
EVICGenesis.pendingDao(uint256,uint256,address) (contracts/distribution/
EVICGenesis.sol#202-217) uses timestamp for comparisons
        Dangerous comparisons:
        - _fromTime >= _toTime (contracts/distribution/EVICGenesis.sol#204)
        - _toTime >= poolEndTime (contracts/distribution/EVICGenesis.sol#205)
        - _toTime <= poolStartTime (contracts/distribution/EVICGenesis.sol#210)
EVICGenesis.updatePool(uint256) (contracts/distribution/EVICGenesis.sol#228-248) uses
timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp <= pool.lastRewardTime (contracts/distribution/
EVICGenesis.sol#230)
EVICGenesis.withdraw(uint256,uint256) (contracts/distribution/EVICGenesis.sol#273-310)
uses timestamp for comparisons
        Dangerous comparisons:
        - _airdropReward > 0 (contracts/distribution/EVICGenesis.sol#284)
        - _daoReward > 0 (contracts/distribution/EVICGenesis.sol#289)
        - _reward > 0 (contracts/distribution/EVICGenesis.sol#298)
EVICGenesis.safeEVICTokenTransfer(address,uint256) (contracts/distribution/
EVICGenesis.sol#313-322) uses timestamp for comparisons
        Dangerous comparisons:
        - _amount > _evicTokenBalance (contracts/distribution/EVICGenesis.sol#316)
EVICGenesis.setPoolStartTime(uint256) (contracts/distribution/EVICGenesis.sol#349-356)
uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp < _poolStartTime,late) (contracts/
distribution/EVICGenesis.sol#350)
        - require(bool,string)(block.timestamp < poolStartTime,Pool is started. Not
reset set time start) (contracts/distribution/EVICGenesis.sol#351)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp

EVICToken.revokeRebaseExclusion(address) (contracts/EVICToken.sol#216-229) has costly
operations inside a loop:
        - excluded.pop() (contracts/EVICToken.sol#224)
EVICGenesis.updatePool(uint256) (contracts/distribution/EVICGenesis.sol#228-248) has
costly operations inside a loop:
        - totalAllocPoint = totalAllocPoint.add(pool.allocPoint) (contracts/
distribution/EVICGenesis.sol#240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-
operations-inside-a-loop

Pragma version0.8.13 (contracts/EVICBoardroomV2.sol#3) necessitates a version too
recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.13 (contracts/EVICToken.sol#3) necessitates a version too recent to
be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.13 (contracts/EVICTreasuryV2.sol#3) necessitates a version too recent
to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.13 (contracts/distribution/EVICGenesis.sol#3) necessitates a version
too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.13 (contracts/utils/ContractGuard.sol#3) necessitates a version too
recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.13 (contracts/utils/EmergencyWithdraw.sol#2) necessitates a version
too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.13 (contracts/utils/Epoch.sol#3) necessitates a version too recent to
be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity

Variable EVICTreasuryV2.maxPercentExpansionTier (contracts/EVICTreasuryV2.sol#85) is
too similar to EVICTreasuryV2.minPercentExpansionTier (contracts/EVICTreasuryV2.sol#84)
Variable EVICGenesis.TOTAL_REWARD_POOL_0_NEXT_PHASE (contracts/distribution/
EVICGenesis.sol#58) is too similar to EVICGenesis.TOTAL_REWARD_POOL_1_NEXT_PHASE
(contracts/distribution/EVICGenesis.sol#57)
Variable EVICGenesis.TOTAL_REWARD_POOL_0_NEXT_PHASE (contracts/distribution/
EVICGenesis.sol#58) is too similar to EVICGenesis.TOTAL_REWARD_POOL_2_NEXT_PHASE
(contracts/distribution/EVICGenesis.sol#56)
Variable EVICGenesis.TOTAL_REWARD_POOL_1_NEXT_PHASE (contracts/distribution/
EVICGenesis.sol#57) is too similar to EVICGenesis.TOTAL_REWARD_POOL_2_NEXT_PHASE
(contracts/distribution/EVICGenesis.sol#56)
Variable EVICGenesis.TOTAL_REWARD_POOL_0_NEXT_PHASE (contracts/distribution/
EVICGenesis.sol#58) is too similar to EVICGenesis.TOTAL_REWARD_POOL_3_NEXT_PHASE
(contracts/distribution/EVICGenesis.sol#55)
Variable EVICGenesis.TOTAL_REWARD_POOL_1_NEXT_PHASE (contracts/distribution/
EVICGenesis.sol#57) is too similar to EVICGenesis.TOTAL_REWARD_POOL_3_NEXT_PHASE
(contracts/distribution/EVICGenesis.sol#55)
Variable EVICGenesis.TOTAL_REWARD_POOL_2_NEXT_PHASE (contracts/distribution/
EVICGenesis.sol#56) is too similar to EVICGenesis.TOTAL_REWARD_POOL_3_NEXT_PHASE
(contracts/distribution/EVICGenesis.sol#55)
Variable EVICGenesis.TOTAL_REWARD_POOL_0_NEXT_PHASE (contracts/distribution/
EVICGenesis.sol#58) is too similar to EVICGenesis.TOTAL_REWARD_POOL_4_NEXT_PHASE
(contracts/distribution/EVICGenesis.sol#54)
Variable EVICGenesis.TOTAL_REWARD_POOL_1_NEXT_PHASE (contracts/distribution/
EVICGenesis.sol#57) is too similar to EVICGenesis.TOTAL_REWARD_POOL_4_NEXT_PHASE
(contracts/distribution/EVICGenesis.sol#54)
Variable EVICGenesis.TOTAL_REWARD_POOL_2_NEXT_PHASE (contracts/distribution/
EVICGenesis.sol#56) is too similar to EVICGenesis.TOTAL_REWARD_POOL_4_NEXT_PHASE
(contracts/distribution/EVICGenesis.sol#54)
Variable EVICGenesis.TOTAL_REWARD_POOL_3_NEXT_PHASE (contracts/distribution/
EVICGenesis.sol#55) is too similar to EVICGenesis.TOTAL_REWARD_POOL_4_NEXT_PHASE
(contracts/distribution/EVICGenesis.sol#54)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-
are-too-similar

setBoardroomAllocPoint(uint256,uint256) should be declared external:
        - EVICTreasuryV2.setBoardroomAllocPoint(uint256,uint256) (contracts/
EVICTreasuryV2.sol#241-246)
getCurrentEpoch() should be declared external:
        - Epoch.getCurrentEpoch() (contracts/utils/Epoch.sol#57-59)
getPeriod() should be declared external:
        - Epoch.getPeriod() (contracts/utils/Epoch.sol#61-63)
getStartTime() should be declared external:
        - Epoch.getStartTime() (contracts/utils/Epoch.sol#65-67)
getLastEpochTime() should be declared external:
        - Epoch.getLastEpochTime() (contracts/utils/Epoch.sol#69-71)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external
```

x Guard