Stader Labs - MaticX

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **April 3rd, 2022 - April 26th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	04/03/2022	Roberto Reigada
0.2	Document Updates	04/05/2022	Roberto Reigada
0.3	Draft Review	04/26/2022	Gabi Urrutia
1.0	Remediation Plan	04/28/2022	Roberto Reigada
1.1	Remediation Plan Review	04/28/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Roberto Reigada	Halborn	Roberto.Reigada@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Stader Labs engaged Halborn to conduct a security audit on their MaticX smart contracts beginning on April 3rd, 2022 and ending on April 5th, 2022. The security assessment was scoped to the smart contract provided in the GitHub repository stader-labs/maticX.

1.2 AUDIT SUMMARY

The team at Halborn was provided a week for the engagement and assigned a full-time security engineer to audit the security of the smart contract. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Ensure that smart contract functions operate as intended
- Identify potential security issues with the smart contracts

In summary, Halborn identified some security risks that were mostly addressed by Stader Labs team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the bridge code and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose
- Smart contract manual code review and walkthrough
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes
- Manual testing by custom scripts
- Scanning of solidity files for vulnerabilities, security hot-spots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Brownie, Remix IDE)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.

3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

IN-SCOPE:

The security assessment was scoped to the following smart contracts:

- MaticX.sol
- ValidatorRegistry.sol
- FxStateChildTunnel.sol
- FxStateRootTunnel.sol
- RateProvider.sol

Commit ID 1:

- eb9f87e2ac124d999b4066a6aada78b71cf701c8

Commit ID 2:

- 8f914608ae40fdb35cfae281ff6c1dda9943b632

Commit ID 3:

- 5ac965782854874d7530203225167b230d893bce

Fixed Commit ID:

- 0c612d147cb11268d168bd4e6eac1ba6608025b4

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	2	0	4

LIKELIHOOD

IMPACT

🟨 (HAL-01) (HAL-02)	🟨	🟥	🟥	🟪
🟩	🟨	🟨	🟥	🟥
🟩	🟩	🟨	🟨	🟥
⬜	🟩	🟩	🟨	🟨
⬜ (HAL-03) (HAL-04) (HAL-05) (HAL-06)	⬜	🟩	🟩	🟨

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - POSSIBLE DENIAL OF SERVICE IN FXSTATECHILDTUNNEL.GETRATE FUNCTION	Medium	SOLVED - 04/28/2022
HAL02 - MISSING REQUIRE STATEMENT IN SETFEEPERCENT	Medium	SOLVED - 04/13/2022
HAL03 - UNNEEDED INITIALIZATION OF UINT256 VARIABLES TO 0	Informational	ACKNOWLEDGED
HAL04 - USING ++I CONSUMES LESS GAS THAN I++ IN LOOPS	Informational	SOLVED - 04/13/2022
HAL05 - PROPOSEDMANAGER STATE VARIABLE CAN BE REMOVED	Informational	SOLVED - 04/13/2022
HAL06 - BOOLEAN EQUALITIES	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) POSSIBLE DENIAL OF SERVICE IN FXSTATECHILDTUNNEL.GETRATE FUNCTION - MEDIUM

Description:

In the MaticX contract, in the requestWithdraw() function, the following call to IFxStateRootTunnel(fxStateRootTunnel).sendMessageToChild() is done:

Listing 1: MaticX.sol (Lines 287-292)

```
function requestWithdraw(uint256 _amount) external override whenNotPaused {
    require(_amount > 0, "Invalid amount");

    (
        uint256 totalAmount2WithdrawInMatic,
        uint256 totalShares,
        uint256 totalPooledMatic
    ) = convertMaticXToMatic(_amount);

    _burn(msg.sender, _amount);

    uint256 leftAmount2WithdrawInMatic = totalAmount2WithdrawInMatic;
    uint256 totalDelegated = getTotalStakeAcrossAllValidators();

    require(
        totalDelegated >= totalAmount2WithdrawInMatic,
        "Too much to withdraw"
    );

    uint256[] memory validators = IValidatorRegistry(validatorRegistry)
        .getValidators();
    uint256 preferredValidatorId = IValidatorRegistry(validatorRegistry)
        .preferredWithdrawalValidatorId();
    uint256 currentIdx = 0;
    for (; currentIdx < validators.length; ++currentIdx) {
        if (preferredValidatorId == validators[currentIdx]) break;
    }

    while (leftAmount2WithdrawInMatic > 0) {
        uint256 validatorId = validators[currentIdx];

        address validatorShare = IStakeManager(stakeManager)
            .getValidatorContract(validatorId);
        (uint256 validatorBalance, ) = getTotalStake(
            IValidatorShare(validatorShare)
        );

        uint256 amount2WithdrawFromValidator = (validatorBalance <=
            leftAmount2WithdrawInMatic)
            ? validatorBalance
            : leftAmount2WithdrawInMatic;

        IValidatorShare(validatorShare).sellVoucher_new(
            amount2WithdrawFromValidator,
            type(uint256).max
        );

        userWithdrawalRequests[msg.sender].push(
            WithdrawalRequest(
                IValidatorShare(validatorShare).unbondNonces(address(this)),
                IStakeManager(stakeManager).epoch() +
                    IStakeManager(stakeManager).withdrawalDelay(),
                validatorShare
            )
        );

        leftAmount2WithdrawInMatic -= amount2WithdrawFromValidator;
        currentIdx = currentIdx + 1 < validators.length
            ? currentIdx + 1
            : 0;
    }

    IFxStateRootTunnel(fxStateRootTunnel).sendMessageToChild(
        abi.encode(
289             totalShares - _amount,
290             totalPooledMatic - totalAmount2WithdrawInMatic
291         )
292     );
293 
294     emit RequestWithdraw(msg.sender, _amount,
 ↳ totalAmount2WithdrawInMatic);
295 }
```

In case totalShares - _amount equals to 0, any calls to the FxStateChildTunnel.getRate() function would revert as the function would try to perform a division by 0:

Listing 2: MaticX.sol (Line 53)

```
42 function getReserves() public view returns (uint256, uint256) {
43     (uint256 maticX, uint256 MATIC) = abi.decode(
44         latestData,
45         (uint256, uint256)
46     );
47 
48     return (maticX, MATIC);
49 }
50 
51 function getRate() external view returns (uint256) {
52     (uint256 maticX, uint256 matic) = getReserves();
53     return (matic * 1 ether) / maticX;
54 }
```

This would cause a Denial of Service on all functions that use the getRate() function.

Risk Level:

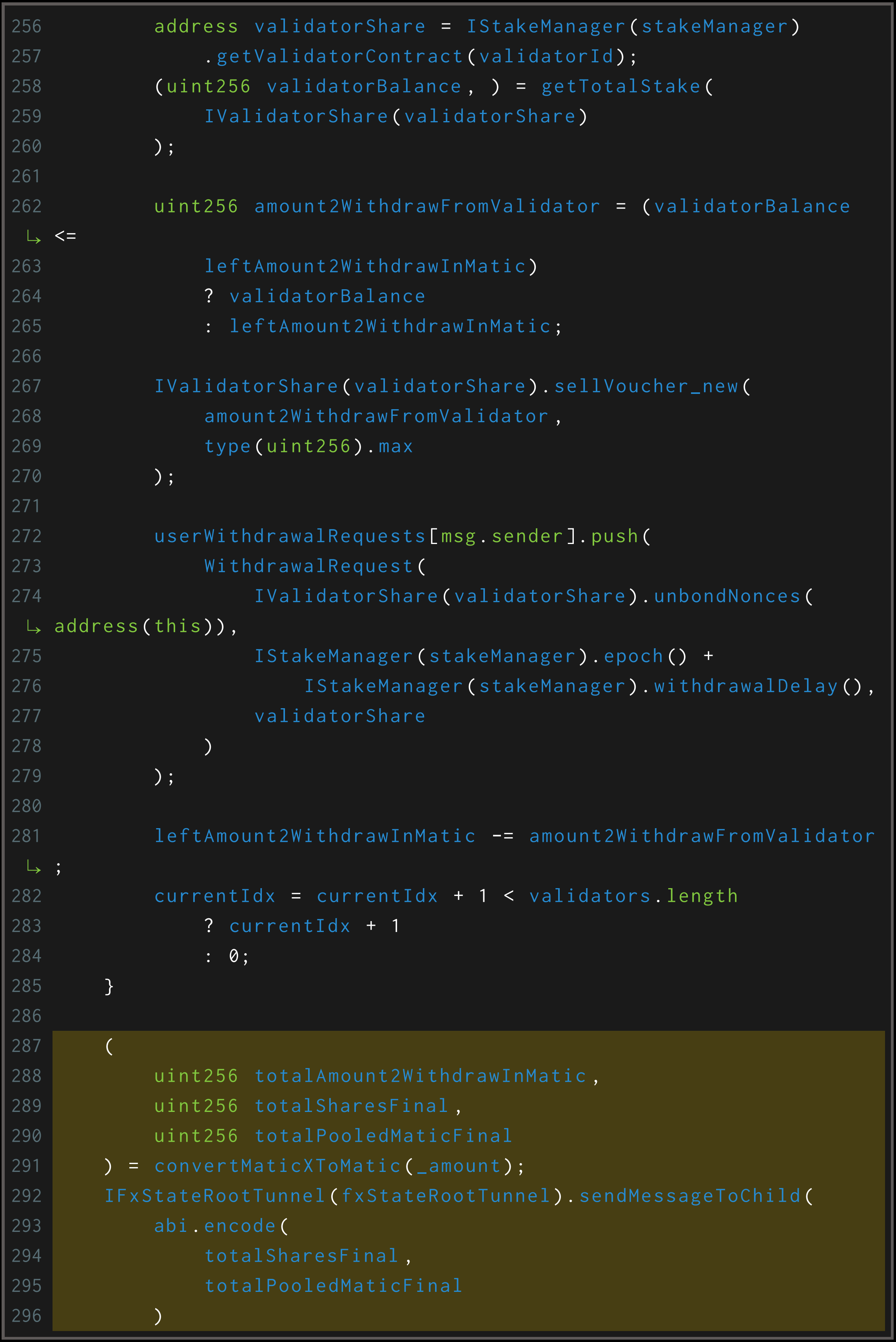
Likelihood - 1
Impact - 5

Recommendation:

It is recommended to call the convertMaticToMaticX() function before the sendMessageToChild() call. convertMaticToMaticX() now correctly handles the edge case where MaticX’s totalSupply or totalPooledMatic is 0. For example:

Listing 3: MaticX.sol (Lines 287-296)

```
function requestWithdraw(uint256 _amount) external override whenNotPaused {
    require(_amount > 0, "Invalid amount");

    (
        uint256 totalAmount2WithdrawInMatic,
        uint256 totalShares,
        uint256 totalPooledMatic
    ) = convertMaticXToMatic(_amount);

    _burn(msg.sender, _amount);

    uint256 leftAmount2WithdrawInMatic = totalAmount2WithdrawInMatic;
    uint256 totalDelegated = getTotalStakeAcrossAllValidators();

    require(
        totalDelegated >= totalAmount2WithdrawInMatic,
        "Too much to withdraw"
    );

    uint256[] memory validators = IValidatorRegistry(validatorRegistry)
        .getValidators();
    uint256 preferredValidatorId = IValidatorRegistry(validatorRegistry)
        .preferredWithdrawalValidatorId();
    uint256 currentIdx = 0;
    for (; currentIdx < validators.length; ++currentIdx) {
        if (preferredValidatorId == validators[currentIdx]) break;
    }

    while (leftAmount2WithdrawInMatic > 0) {
        uint256 validatorId = validators[currentIdx];

```



```
        address validatorShare = IStakeManager(stakeManager)
            .getValidatorContract(validatorId);
        (uint256 validatorBalance, ) = getTotalStake(
            IValidatorShare(validatorShare)
        );

        uint256 amount2WithdrawFromValidator = (validatorBalance <=
            leftAmount2WithdrawInMatic)
            ? validatorBalance
            : leftAmount2WithdrawInMatic;

        IValidatorShare(validatorShare).sellVoucher_new(
            amount2WithdrawFromValidator,
            type(uint256).max
        );

        userWithdrawalRequests[msg.sender].push(
            WithdrawalRequest(
                IValidatorShare(validatorShare).unbondNonces(address(this)),
                IStakeManager(stakeManager).epoch() +
                    IStakeManager(stakeManager).withdrawalDelay(),
                validatorShare
            )
        );

        leftAmount2WithdrawInMatic -= amount2WithdrawFromValidator;
        currentIdx = currentIdx + 1 < validators.length
            ? currentIdx + 1
            : 0;
    }

    (
        uint256 totalAmount2WithdrawInMatic,
        uint256 totalSharesFinal,
        uint256 totalPooledMaticFinal
    ) = convertMaticXToMatic(_amount);
    IFxStateRootTunnel(fxStateRootTunnel).sendMessageToChild(
        abi.encode(
            totalSharesFinal,
            totalPooledMaticFinal
        )
    );

    emit RequestWithdraw(msg.sender, _amount, totalAmount2WithdrawInMatic);
}
```

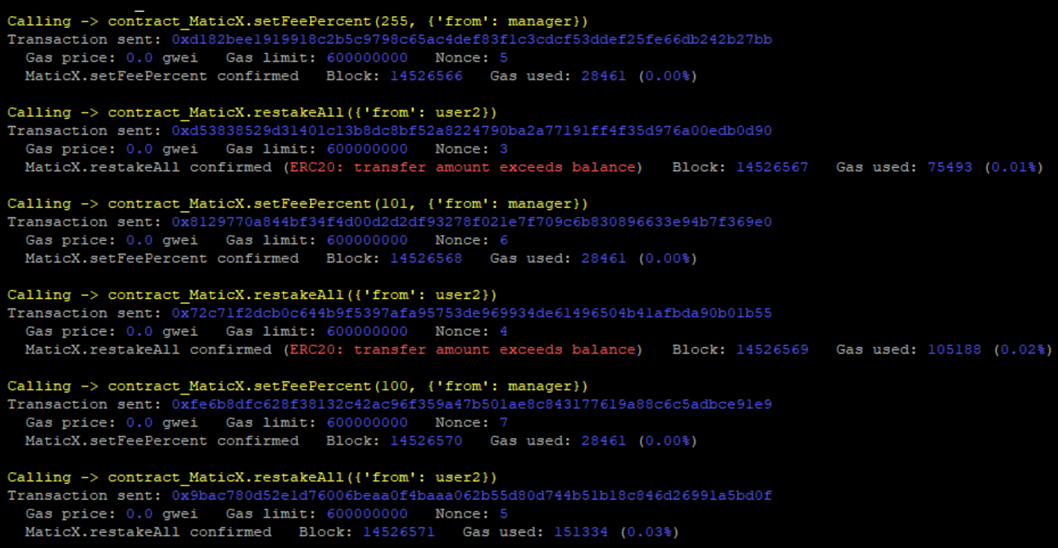
Remediation Plan:

SOLVED: The Stader Labs team fixed the issue. The FxStateChildTunnel.getRate() function now makes use of the convertMaticXToMatic() function in the FxStateChildTunnel contract that handles the edge case where Matic or MaticX is equal to 0.

3.2 (HAL-02) MISSING REQUIRE STATEMENT IN SETFEEPERCENT - MEDIUM

Description:

In the MaticX contract, the function setFeePercent() is missing a require statement that restricts the feePercent setting to a value greater than 100. Setting feePercent to a value higher than 100 would cause users to not they could for rewards, as it would be impossible to re-stake any validator with rewards:



```
Calling -> contract_MaticX.setFeePercent(255, {'from': manager})
Transaction sent: 0xd182bee1919918c2b5c9798c65ac4def83f1c3cdcf53ddef25fe66db242b27bb
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 5
  MaticX.setFeePercent confirmed   Block: 14526566   Gas used: 28461 (0.00%)

Calling -> contract_MaticX.restakeAll({'from': user2})
Transaction sent: 0xd53838529d31401c13b8dc8bf52a8224790ba2a77191ff4f35d976a00edb0d90
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 3
  MaticX.restakeAll confirmed (ERC20: transfer amount exceeds balance)   Block: 14526567   Gas used: 75493 (0.01%)

Calling -> contract_MaticX.setFeePercent(101, {'from': manager})
Transaction sent: 0x8129770a844bf34f4d00d2d2df93278f021e7f709c6b830896633e94b7f369e0
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 6
  MaticX.setFeePercent confirmed   Block: 14526568   Gas used: 28461 (0.00%)

Calling -> contract_MaticX.restakeAll({'from': user2})
Transaction sent: 0x72c71f2dcb0c644b9f5397afa95753de969934de61496504b41afbda90b01b55
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 4
  MaticX.restakeAll confirmed (ERC20: transfer amount exceeds balance)   Block: 14526569   Gas used: 105188 (0.02%)

Calling -> contract_MaticX.setFeePercent(100, {'from': manager})
Transaction sent: 0xfe6b8dfc628f38132c42ac96f359a47b501ae8c843177619a88c6c5adbce91e9
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 7
  MaticX.setFeePercent confirmed   Block: 14526570   Gas used: 28461 (0.00%)

Calling -> contract_MaticX.restakeAll({'from': user2})
Transaction sent: 0x9bac780d52e1d76006beaa0f4baaa062b55d80d744b51b18c846d26991a5bd0f
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 5
  MaticX.restakeAll confirmed   Block: 14526571   Gas used: 151334 (0.03%)
```

Code Location:

Listing 4: MaticX.sol (Line 629)

```
624     function setFeePercent(uint8 _feePercent)
625         external
626         override
627         onlyRole(DEFAULT_ADMIN_ROLE)
628     {
629         feePercent = _feePercent;
630     }
```

Risk Level:

Likelihood - 1
Impact - 5

Recommendation:

It is recommended to add the following require statement in the setFeePercent() function.

```
require(_feePercent <= 100, ”_feePercent must be <= 100”);
```

Also consider limiting it to, for example, 10-20%. In this way, users will always be sure that their rewards will be reduced by a maximum of that amount.

Remediation Plan:

SOLVED: The Stader Labs team added the suggested require statement to the setFeePercent() function.

3.3 (HAL-03) UNNEEDED INITIALIZATION OF UINT256 VARIABLES TO 0 - INFORMATIONAL

Description:

uint256 variables are already initialized to 0 by default. uint256 i = 0 would reassign the 0 to i which wastes gas.

Code Location:

MaticX.sol

- Line 252: `uint256 currentIdx = 0;`
- Line 311: `for (uint256 idx = 0; idx < validators.length; idx++){`
- Line 511: `uint256 amountToClaim = 0;`
- Line 709: `for (uint256 i = 0; i < validators.length; i++){`

ValidatorRegistry.sol

- Line 119: `for (uint256 idx = 0; idx < validators.length - 1; idx++){`

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended not to initialize uint256 variables to 0 to save gas. For example, use instead: `for (uint256 idx; idx < validators.length; ++idx){`.

Remediation Plan:

ACKNOWLEDGED: The Stader Labs team acknowledged this finding.

3.4 (HAL-04) USING ++I CONSUMES LESS GAS THAN I++ IN LOOPS - INFORMATIONAL

Description:

In the loop below, the variable i is incremented using i++. It is known that, in loops, using ++i costs less gas per iteration than i++.

Code Location:

MaticX.sol

- Line 253: for (; currentIdx < validators.length; currentIdx++){
- Line 311: for (uint256 idx = 0; idx < validators.length; idx++){
- Line 709: for (uint256 i = 0; i < validators.length; i++){

ValidatorRegistry.sol

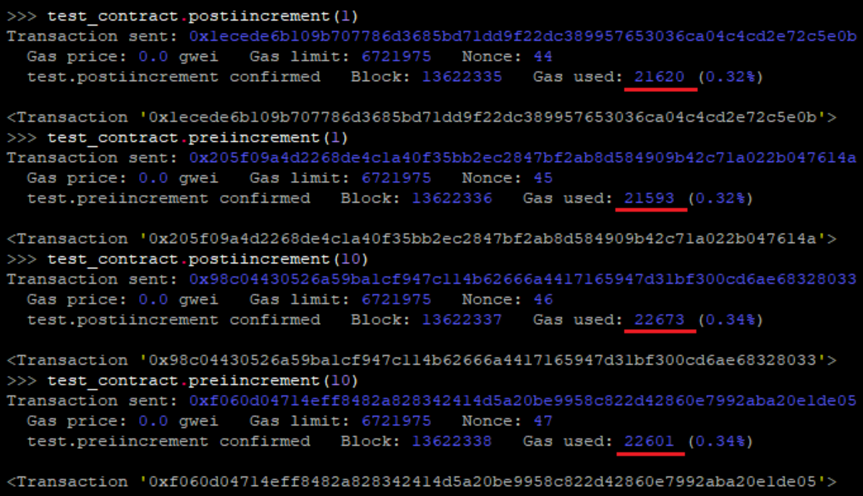
- Line 119: for (uint256 idx = 0; idx < validators.length - 1; idx++){

Proof of Concept:

For example, based on the following test contract:

Listing 5: Test.sol

```
//SPDX-License-Identifier: MIT
pragma solidity 0.8.9;

contract test {
    function postiincrement(uint256 iterations) public {
        for (uint256 i = 0; i < iterations; i++) {
        }
    }
    function preiincrement(uint256 iterations) public {
        for (uint256 i = 0; i < iterations; ++i) {
        }
    }
}
```



```
>>> test_contract.postiincrement(1)
Transaction sent: 0x1ecede6b109b707786d3685bd71dd9f22dc389957653036ca04c4cd2e72c5e0b
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 44
  test.postiincrement confirmed   Block: 13622335   Gas used: 21620 (0.32%)

<Transaction '0x1ecede6b109b707786d3685bd71dd9f22dc389957653036ca04c4cd2e72c5e0b'>
>>> test_contract.preiincrement(1)
Transaction sent: 0x205f09a4d2268de4c1a40f35bb2ec2847bf2ab8d584909b42c71a022b047614a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 45
  test.preiincrement confirmed   Block: 13622336   Gas used: 21593 (0.32%)

<Transaction '0x205f09a4d2268de4c1a40f35bb2ec2847bf2ab8d584909b42c71a022b047614a'>
>>> test_contract.postiincrement(10)
Transaction sent: 0x98c04430526a59ba1cf947c114b62666a4417165947d31bf300cd6ae68328033
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 46
  test.postiincrement confirmed   Block: 13622337   Gas used: 22673 (0.34%)

<Transaction '0x98c04430526a59ba1cf947c114b62666a4417165947d31bf300cd6ae68328033'>
>>> test_contract.preiincrement(10)
Transaction sent: 0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20e1de05
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 47
  test.preiincrement confirmed   Block: 13622338   Gas used: 22601 (0.34%)

<Transaction '0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20e1de05'>
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to use ++i instead of i++ to increment the value of a uint variable within a loop. This does not just apply to the iterator variable. It also applies to increments made within the loop code block.

Remediation Plan:

SOLVED: The Stader Labs team now uses ++i in the for loops to increase the iterator variable, reducing the gas costs.

3.5 (HAL-05) PROPOSEDMANAGER STATE VARIABLE CAN BE REMOVED - INFORMATIONAL

Description:

In the contract MaticX, the proposed_manager state variable is declared, but it is not used anywhere in the smart contract.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to remove the proposed_manager state variable from the MaticX contract.

Remediation Plan:

SOLVED: The Stader Labs team removed the proposed_manager state variable.

3.6 (HAL-06) BOOLEAN EQUALITIES - INFORMATIONAL

Description:

Boolean constants can be used directly and do not need to be compared to true or false.

Code Location:

Listing 6: ValidatorRegistry.sol (Line 262)

```
260    modifier whenValidatorIdExists(uint256 _validatorId) {
261        require(
262            validatorIdExists[_validatorId] == true,
263            "Validator id doesn't exist in our registry"
264        );
265        _;
266    }
```

Listing 7: ValidatorRegistry.sol (Line 277)

```
275    modifier whenValidatorIdDoesNotExist(uint256 _validatorId) {
276        require(
277            validatorIdExists[_validatorId] == false,
278            "Validator id already exists in our registry"
279        );
280        _;
281    }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to remove the equality to the boolean constant, for example:

Listing 8: ValidatorRegistry.sol (Line 262)

```
260     modifier whenValidatorIdExists(uint256 _validatorId) {
261         require(
262             validatorIdExists[_validatorId],
263             "Validator id doesn't exist in our registry"
264         );
265         _;
266     }
```

Listing 9: ValidatorRegistry.sol (Line 277)

```
275     modifier whenValidatorIdDoesNotExist(uint256 _validatorId) {
276         require(
277             !validatorIdExists[_validatorId],
278             "Validator id already exists in our registry"
279         );
280         _;
281     }
```

Remediation Plan:

ACKNOWLEDGED: The Stader Labs team acknowledged this finding.

AUTOMATED TESTING

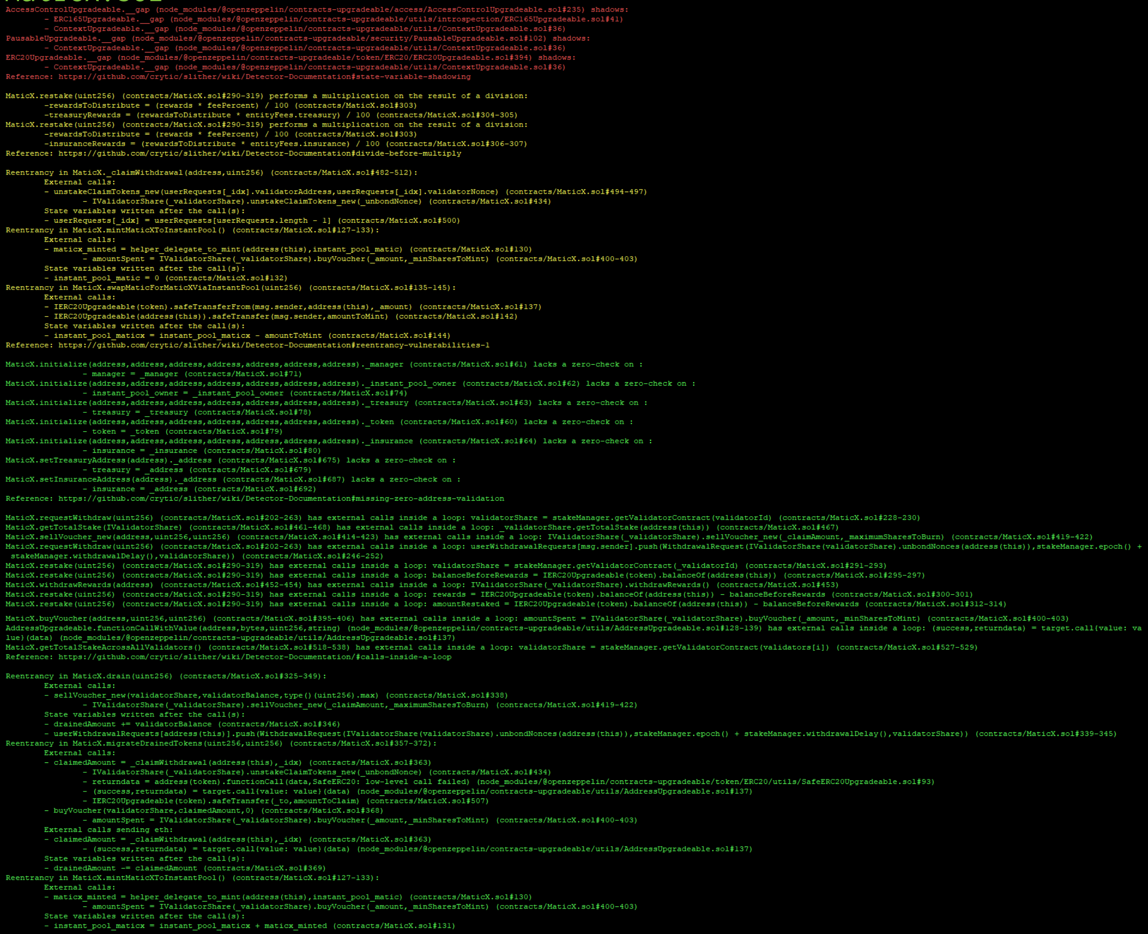
4.1 STATIC ANALYSIS REPORT

Description:

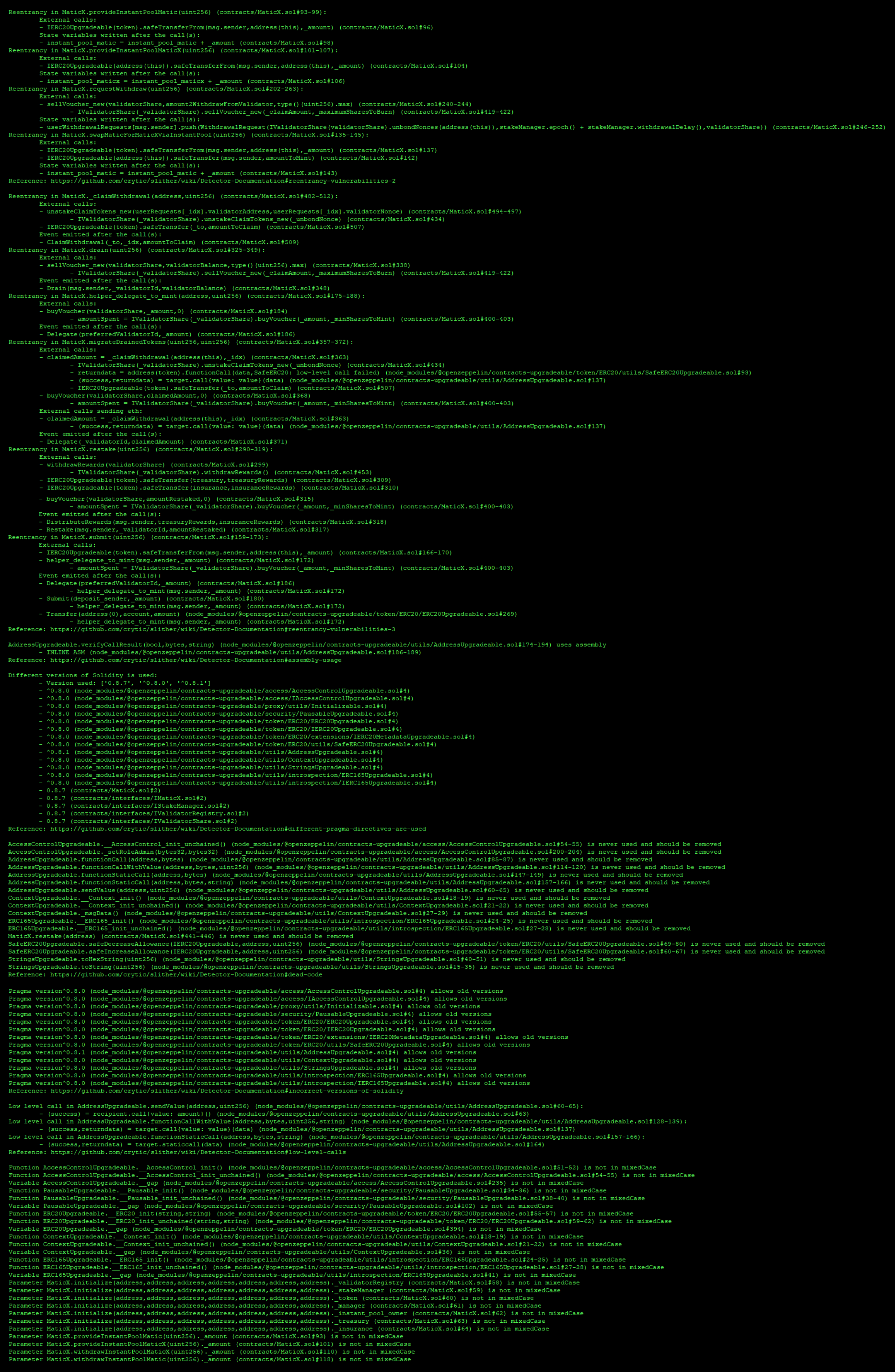
Halborn used automated testing techniques to enhance the coverage of certain areas of the smart contract in scope. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified the smart contract in the repository and was able to compile it correctly into its abi and binary format, Slither was run against the contract. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Slither results:

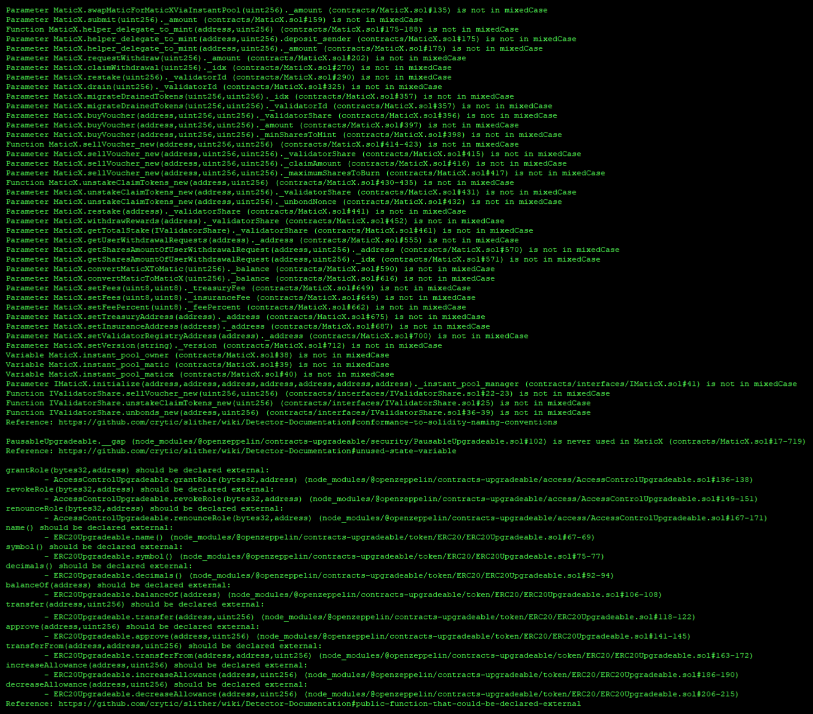
MaticX.sol



```
AccessControlUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#235) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#41)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#36)
PausableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#102) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#36)
ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#394) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

MaticX.restake(uint256) (contracts/MaticX.sol#290-319) performs a multiplication on the result of a division:
        -rewardsToDistribute = (rewards * feePercent) / 100 (contracts/MaticX.sol#303)
        -treasuryRewards = (rewardsToDistribute * entityFees.treasury) / 100 (contracts/MaticX.sol#304-305)
MaticX.restake(uint256) (contracts/MaticX.sol#290-319) performs a multiplication on the result of a division:
        -rewardsToDistribute = (rewards * feePercent) / 100 (contracts/MaticX.sol#303)
        -insuranceRewards = (rewardsToDistribute * entityFees.insurance) / 100 (contracts/MaticX.sol#306-307)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Reentrancy in MaticX._claimWithdrawal(address,uint256) (contracts/MaticX.sol#482-512):
        External calls:
        - unstakeClaimTokens_new(userRequests[_idx].validatorAddress,userRequests[_idx].validatorNonce) (contracts/MaticX.sol#494-497)
                - IValidatorShare(_validatorShare).unstakeClaimTokens_new(_unbondNonce) (contracts/MaticX.sol#434)
        State variables written after the call(s):
        - userRequests[_idx] = userRequests[userRequests.length - 1] (contracts/MaticX.sol#500)
Reentrancy in MaticX.mintMaticXToInstantPool() (contracts/MaticX.sol#127-133):
        External calls:
        - maticx_minted = helper_delegate_to_mint(address(this),instant_pool_matic) (contracts/MaticX.sol#130)
                - amountSpent = IValidatorShare(_validatorShare).buyVoucher(_amount,_minSharesToMint) (contracts/MaticX.sol#400-403)
        State variables written after the call(s):
        - instant_pool_matic = 0 (contracts/MaticX.sol#132)
Reentrancy in MaticX.swapMaticForMaticXViaInstantPool(uint256) (contracts/MaticX.sol#135-145):
        External calls:
        - IERC20Upgradeable(token).safeTransferFrom(msg.sender,address(this),_amount) (contracts/MaticX.sol#137)
        - IERC20Upgradeable(address(this)).safeTransfer(msg.sender,amountToMint) (contracts/MaticX.sol#142)
        State variables written after the call(s):
        - instant_pool_maticx = instant_pool_maticx - amountToMint (contracts/MaticX.sol#144)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

MaticX.initialize(address,address,address,address,address,address,address)._manager (contracts/MaticX.sol#61) lacks a zero-check on :
                - manager = _manager (contracts/MaticX.sol#71)
MaticX.initialize(address,address,address,address,address,address,address)._instant_pool_owner (contracts/MaticX.sol#62) lacks a zero-check on :
                - instant_pool_owner = _instant_pool_owner (contracts/MaticX.sol#74)
MaticX.initialize(address,address,address,address,address,address,address)._treasury (contracts/MaticX.sol#63) lacks a zero-check on :
                - treasury = _treasury (contracts/MaticX.sol#78)
MaticX.initialize(address,address,address,address,address,address,address)._token (contracts/MaticX.sol#60) lacks a zero-check on :
                - token = _token (contracts/MaticX.sol#79)
MaticX.initialize(address,address,address,address,address,address,address)._insurance (contracts/MaticX.sol#64) lacks a zero-check on :
                - insurance = _insurance (contracts/MaticX.sol#80)
MaticX.setTreasuryAddress(address)._address (contracts/MaticX.sol#675) lacks a zero-check on :
                - treasury = _address (contracts/MaticX.sol#679)
MaticX.setInsuranceAddress(address)._address (contracts/MaticX.sol#687) lacks a zero-check on :
                - insurance = _address (contracts/MaticX.sol#692)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

MaticX.requestWithdraw(uint256) (contracts/MaticX.sol#202-263) has external calls inside a loop: validatorShare = stakeManager.getValidatorContract(validatorId) (contracts/MaticX.sol#228-230)
MaticX.getTotalStake(IValidatorShare) (contracts/MaticX.sol#461-468) has external calls inside a loop: _validatorShare.getTotalStake(address(this)) (contracts/MaticX.sol#467)
MaticX.sellVoucher_new(address,uint256,uint256) (contracts/MaticX.sol#414-423) has external calls inside a loop: IValidatorShare(_validatorShare).sellVoucher_new(_claimAmount,_maximumSharesToBurn) (contracts/MaticX.sol#419-422)
MaticX.requestWithdraw(uint256) (contracts/MaticX.sol#202-263) has external calls inside a loop: userWithdrawalRequests[msg.sender].push(WithdrawalRequest(IValidatorShare(validatorShare).unbondNonces(address(this)),stakeManager.epoch() +
 stakeManager.withdrawalDelay(),validatorShare)) (contracts/MaticX.sol#246-252)
MaticX.restake(uint256) (contracts/MaticX.sol#290-319) has external calls inside a loop: validatorShare = stakeManager.getValidatorContract(_validatorId) (contracts/MaticX.sol#291-293)
MaticX.restake(uint256) (contracts/MaticX.sol#290-319) has external calls inside a loop: balanceBeforeRewards = IERC20Upgradeable(token).balanceOf(address(this)) (contracts/MaticX.sol#295-297)
MaticX.withdrawRewards(address) (contracts/MaticX.sol#452-454) has external calls inside a loop: IValidatorShare(_validatorShare).withdrawRewards() (contracts/MaticX.sol#453)
MaticX.restake(uint256) (contracts/MaticX.sol#290-319) has external calls inside a loop: rewards = IERC20Upgradeable(token).balanceOf(address(this)) - balanceBeforeRewards (contracts/MaticX.sol#300-301)
MaticX.restake(uint256) (contracts/MaticX.sol#290-319) has external calls inside a loop: amountRestaked = IERC20Upgradeable(token).balanceOf(address(this)) - balanceBeforeRewards (contracts/MaticX.sol#312-314)

MaticX.buyVoucher(address,uint256,uint256) (contracts/MaticX.sol#395-406) has external calls inside a loop: amountSpent = IValidatorShare(_validatorShare).buyVoucher(_amount,_minSharesToMint) (contracts/MaticX.sol#400-403)
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139) has external calls inside a loop: (success,returndata) = target.call{value: va
lue}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#137)
MaticX.getTotalStakeAcrossAllValidators() (contracts/MaticX.sol#518-538) has external calls inside a loop: validatorShare = stakeManager.getValidatorContract(validators[i]) (contracts/MaticX.sol#527-529)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in MaticX.drain(uint256) (contracts/MaticX.sol#325-349):
        External calls:
        - sellVoucher_new(validatorShare,validatorBalance,type()(uint256).max) (contracts/MaticX.sol#338)
                - IValidatorShare(_validatorShare).sellVoucher_new(_claimAmount,_maximumSharesToBurn) (contracts/MaticX.sol#419-422)
        State variables written after the call(s):
        - drainedAmount += validatorBalance (contracts/MaticX.sol#346)
        - userWithdrawalRequests[address(this)].push(WithdrawalRequest(IValidatorShare(validatorShare).unbondNonces(address(this)),stakeManager.epoch() + stakeManager.withdrawalDelay(),validatorShare)) (contracts/MaticX.sol#339-345)
Reentrancy in MaticX.migrateDrainedTokens(uint256,uint256) (contracts/MaticX.sol#357-372):
        External calls:
        - claimedAmount = _claimWithdrawal(address(this),_idx) (contracts/MaticX.sol#363)
                - IValidatorShare(_validatorShare).unstakeClaimTokens_new(_unbondNonce) (contracts/MaticX.sol#434)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#93)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#137)
                - IERC20Upgradeable(token).safeTransfer(_to,amountToClaim) (contracts/MaticX.sol#507)
        - buyVoucher(validatorShare,claimedAmount,0) (contracts/MaticX.sol#368)
                - amountSpent = IValidatorShare(_validatorShare).buyVoucher(_amount,_minSharesToMint) (contracts/MaticX.sol#400-403)
        External calls sending eth:
        - claimedAmount = _claimWithdrawal(address(this),_idx) (contracts/MaticX.sol#363)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#137)
        State variables written after the call(s):
        - drainedAmount -= claimedAmount (contracts/MaticX.sol#369)
Reentrancy in MaticX.mintMaticXToInstantPool() (contracts/MaticX.sol#127-133):
        External calls:
        - maticx_minted = helper_delegate_to_mint(address(this),instant_pool_matic) (contracts/MaticX.sol#130)
                - amountSpent = IValidatorShare(_validatorShare).buyVoucher(_amount,_minSharesToMint) (contracts/MaticX.sol#400-403)
        State variables written after the call(s):
        - instant_pool_maticx = instant_pool_maticx + maticx_minted (contracts/MaticX.sol#131)
```

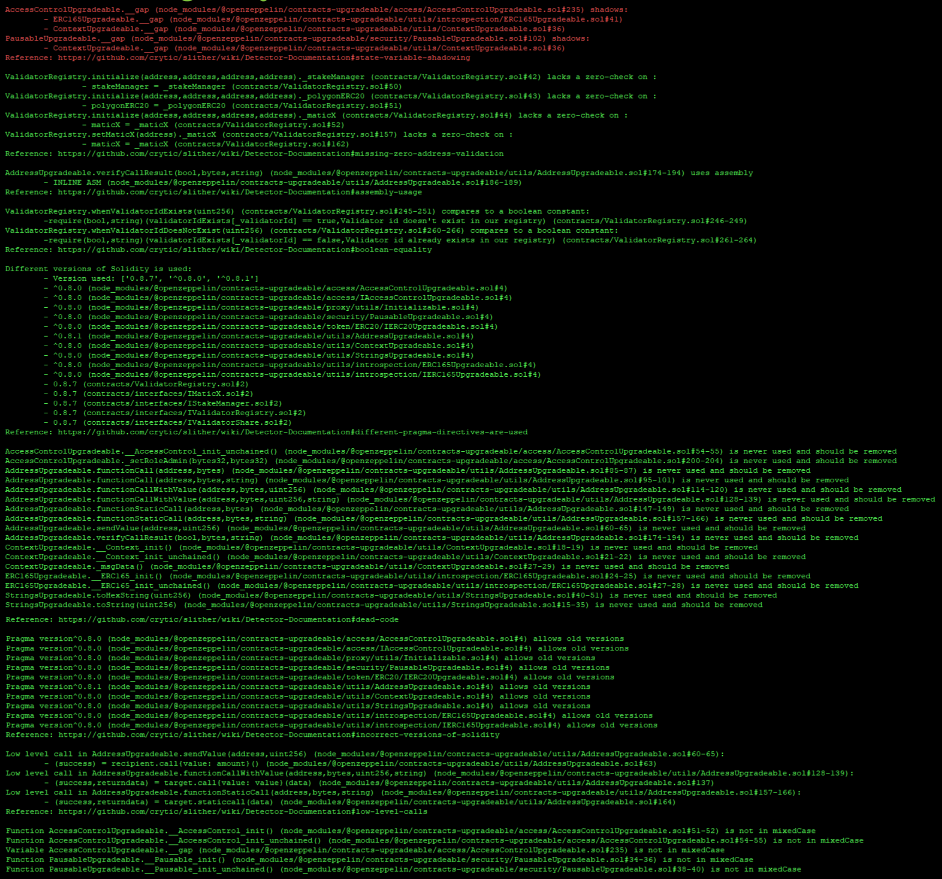


```
Reentrancy in MaticX.provideInstantPoolMatic(uint256) (contracts/MaticX.sol#93-99):
        External calls:
        - IERC20Upgradeable(token).safeTransferFrom(msg.sender,address(this),_amount) (contracts/MaticX.sol#96)
        State variables written after the call(s):
        - instant_pool_matic = instant_pool_matic + _amount (contracts/MaticX.sol#98)
Reentrancy in MaticX.provideInstantPoolMaticX(uint256) (contracts/MaticX.sol#101-107):
        External calls:
        - IERC20Upgradeable(address(this)).safeTransferFrom(msg.sender,address(this),_amount) (contracts/MaticX.sol#104)
        State variables written after the call(s):
        - instant_pool_maticx = instant_pool_maticx + _amount (contracts/MaticX.sol#106)
Reentrancy in MaticX.requestWithdraw(uint256) (contracts/MaticX.sol#202-263):
        External calls:
        - sellVoucher_new(validatorShare,amount2WithdrawFromValidator,type()(uint256).max) (contracts/MaticX.sol#240-244)
                - IValidatorShare(_validatorShare).sellVoucher_new(_claimAmount,_maximumSharesToBurn) (contracts/MaticX.sol#419-422)
        State variables written after the call(s):
        - userWithdrawalRequests[msg.sender].push(WithdrawalRequest(IValidatorShare(validatorShare).unbondNonces(address(this)),stakeManager.epoch() + stakeManager.withdrawalDelay(),validatorShare)) (contracts/MaticX.sol#246-252)
Reentrancy in MaticX.swapMaticForMaticXViaInstantPool(uint256) (contracts/MaticX.sol#135-145):
        External calls:
        - IERC20Upgradeable(token).safeTransferFrom(msg.sender,address(this),_amount) (contracts/MaticX.sol#137)
        - IERC20Upgradeable(address(this)).safeTransfer(msg.sender,amountToMint) (contracts/MaticX.sol#142)
        State variables written after the call(s):
        - instant_pool_matic = instant_pool_matic + _amount (contracts/MaticX.sol#143)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in MaticX._claimWithdrawal(address,uint256) (contracts/MaticX.sol#482-512):
        External calls:
        - unstakeClaimTokens_new(userRequests[_idx].validatorAddress,userRequests[_idx].validatorNonce) (contracts/MaticX.sol#494-497)
                - IValidatorShare(_validatorShare).unstakeClaimTokens_new(_unbondNonce) (contracts/MaticX.sol#434)
        - IERC20Upgradeable(token).safeTransfer(_to,amountToClaim) (contracts/MaticX.sol#507)
        Event emitted after the call(s):
        - ClaimWithdrawal(_to,_idx,amountToClaim) (contracts/MaticX.sol#509)
Reentrancy in MaticX.drain(uint256) (contracts/MaticX.sol#325-349):
        External calls:
        - sellVoucher_new(validatorShare,validatorBalance,type()(uint256).max) (contracts/MaticX.sol#338)
                - IValidatorShare(_validatorShare).sellVoucher_new(_claimAmount,_maximumSharesToBurn) (contracts/MaticX.sol#419-422)
        Event emitted after the call(s):
        - Drain(msg.sender,_validatorId,validatorBalance) (contracts/MaticX.sol#348)
Reentrancy in MaticX.helper_delegate_to_mint(address,uint256) (contracts/MaticX.sol#175-188):
        External calls:
        - buyVoucher(validatorShare,_amount,0) (contracts/MaticX.sol#184)
                - amountSpent = IValidatorShare(_validatorShare).buyVoucher(_amount,_minSharesToMint) (contracts/MaticX.sol#400-403)
        Event emitted after the call(s):
        - Delegate(preferredValidatorId,_amount) (contracts/MaticX.sol#186)
Reentrancy in MaticX.migrateDrainedTokens(uint256,uint256) (contracts/MaticX.sol#357-372):
        External calls:
        - claimedAmount = _claimWithdrawal(address(this),_idx) (contracts/MaticX.sol#363)
                - IValidatorShare(_validatorShare).unstakeClaimTokens_new(_unbondNonce) (contracts/MaticX.sol#434)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#93)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#137)
                - IERC20Upgradeable(token).safeTransfer(_to,amountToClaim) (contracts/MaticX.sol#507)
        - buyVoucher(validatorShare,claimedAmount,0) (contracts/MaticX.sol#368)
                - amountSpent = IValidatorShare(_validatorShare).buyVoucher(_amount,_minSharesToMint) (contracts/MaticX.sol#400-403)
        External calls sending eth:
        - claimedAmount = _claimWithdrawal(address(this),_idx) (contracts/MaticX.sol#363)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#137)
        Event emitted after the call(s):
        - Delegate(_validatorId,claimedAmount) (contracts/MaticX.sol#371)
Reentrancy in MaticX.restake(uint256) (contracts/MaticX.sol#290-319):
        External calls:
        - withdrawRewards(validatorShare) (contracts/MaticX.sol#299)
                - IValidatorShare(_validatorShare).withdrawRewards() (contracts/MaticX.sol#453)
        - IERC20Upgradeable(token).safeTransfer(treasury,treasuryRewards) (contracts/MaticX.sol#309)
        - IERC20Upgradeable(token).safeTransfer(insurance,insuranceRewards) (contracts/MaticX.sol#310)

        - buyVoucher(validatorShare,amountRestaked,0) (contracts/MaticX.sol#315)
                - amountSpent = IValidatorShare(_validatorShare).buyVoucher(_amount,_minSharesToMint) (contracts/MaticX.sol#400-403)
        Event emitted after the call(s):
        - DistributeRewards(msg.sender,treasuryRewards,insuranceRewards) (contracts/MaticX.sol#318)
        - Restake(msg.sender,_validatorId,amountRestaked) (contracts/MaticX.sol#317)
Reentrancy in MaticX.submit(uint256) (contracts/MaticX.sol#159-173):
        External calls:
        - IERC20Upgradeable(token).safeTransferFrom(msg.sender,address(this),_amount) (contracts/MaticX.sol#166-170)
        - helper_delegate_to_mint(msg.sender,_amount) (contracts/MaticX.sol#172)
                - amountSpent = IValidatorShare(_validatorShare).buyVoucher(_amount,_minSharesToMint) (contracts/MaticX.sol#400-403)
        Event emitted after the call(s):
        - Delegate(preferredValidatorId,_amount) (contracts/MaticX.sol#186)
                - helper_delegate_to_mint(msg.sender,_amount) (contracts/MaticX.sol#172)
        - Submit(deposit_sender,_amount) (contracts/MaticX.sol#180)
                - helper_delegate_to_mint(msg.sender,_amount) (contracts/MaticX.sol#172)
        - Transfer(address(0),account,amount) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#269)
                - helper_delegate_to_mint(msg.sender,_amount) (contracts/MaticX.sol#172)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#186-189)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.8.7', '^0.8.0', '^0.8.1']
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/IAccessControlUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol#4)
        - 0.8.7 (contracts/MaticX.sol#2)
        - 0.8.7 (contracts/interfaces/IMaticX.sol#2)
        - 0.8.7 (contracts/interfaces/IStakeManager.sol#2)
        - 0.8.7 (contracts/interfaces/IValidatorRegistry.sol#2)
        - 0.8.7 (contracts/interfaces/IValidatorShare.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

AccessControlUpgradeable.__AccessControl_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#54-55) is never used and should be removed
AccessControlUpgradeable._setRoleAdmin(bytes32,bytes32) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#200-204) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#85-87) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#114-120) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#147-149) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65) is never used and should be removed
ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is never used and should be removed
ContextUpgradeable.__Context_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is never used and should be removed
ContextUpgradeable._msgData() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#27-29) is never used and should be removed
ERC165Upgradeable.__ERC165_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#24-25) is never used and should be removed
ERC165Upgradeable.__ERC165_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#27-28) is never used and should be removed
MaticX.restake(address) (contracts/MaticX.sol#441-446) is never used and should be removed
SafeERC20Upgradeable.safeDecreaseAllowance(IERC20Upgradeable,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#69-80) is never used and should be removed
SafeERC20Upgradeable.safeIncreaseAllowance(IERC20Upgradeable,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#60-67) is never used and should be removed
StringsUpgradeable.toHexString(uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#40-51) is never used and should be removed
StringsUpgradeable.toString(uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#15-35) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/IAccessControlUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol#4) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#137)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#164)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function AccessControlUpgradeable.__AccessControl_init() (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#51-52) is not in mixedCase
Function AccessControlUpgradeable.__AccessControl_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#54-55) is not in mixedCase
Variable AccessControlUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#235) is not in mixedCase
Function PausableUpgradeable.__Pausable_init() (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#34-36) is not in mixedCase
Function PausableUpgradeable.__Pausable_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#38-40) is not in mixedCase
Variable PausableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#102) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init(string,string) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#55-57) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init_unchained(string,string) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#59-62) is not in mixedCase
Variable ERC20Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#394) is not in mixedCase
Function ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is not in mixedCase
Variable ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#36) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#24-25) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#27-28) is not in mixedCase
Variable ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#41) is not in mixedCase
Parameter MaticX.initialize(address,address,address,address,address,address,address)._validatorRegistry (contracts/MaticX.sol#58) is not in mixedCase
Parameter MaticX.initialize(address,address,address,address,address,address,address)._stakeManager (contracts/MaticX.sol#59) is not in mixedCase
Parameter MaticX.initialize(address,address,address,address,address,address,address)._token (contracts/MaticX.sol#60) is not in mixedCase
Parameter MaticX.initialize(address,address,address,address,address,address,address)._manager (contracts/MaticX.sol#61) is not in mixedCase
Parameter MaticX.initialize(address,address,address,address,address,address,address)._instant_pool_owner (contracts/MaticX.sol#62) is not in mixedCase
Parameter MaticX.initialize(address,address,address,address,address,address,address)._treasury (contracts/MaticX.sol#63) is not in mixedCase
Parameter MaticX.initialize(address,address,address,address,address,address,address)._insurance (contracts/MaticX.sol#64) is not in mixedCase
Parameter MaticX.provideInstantPoolMatic(uint256)._amount (contracts/MaticX.sol#93) is not in mixedCase
Parameter MaticX.provideInstantPoolMaticX(uint256)._amount (contracts/MaticX.sol#101) is not in mixedCase
Parameter MaticX.withdrawInstantPoolMaticX(uint256)._amount (contracts/MaticX.sol#110) is not in mixedCase
Parameter MaticX.withdrawInstantPoolMatic(uint256)._amount (contracts/MaticX.sol#118) is not in mixedCase
```



```
Parameter MaticX.swapMaticForMaticXViaInstantPool(uint256)._amount (contracts/MaticX.sol#135) is not in mixedCase
Parameter MaticX.submit(uint256)._amount (contracts/MaticX.sol#159) is not in mixedCase
Function MaticX.helper_delegate_to_mint(address,uint256) (contracts/MaticX.sol#175-188) is not in mixedCase
Parameter MaticX.helper_delegate_to_mint(address,uint256).deposit_sender (contracts/MaticX.sol#175) is not in mixedCase
Parameter MaticX.helper_delegate_to_mint(address,uint256)._amount (contracts/MaticX.sol#175) is not in mixedCase
Parameter MaticX.requestWithdraw(uint256)._amount (contracts/MaticX.sol#202) is not in mixedCase
Parameter MaticX.claimWithdrawal(uint256)._idx (contracts/MaticX.sol#270) is not in mixedCase
Parameter MaticX.restake(uint256)._validatorId (contracts/MaticX.sol#290) is not in mixedCase
Parameter MaticX.drain(uint256)._validatorId (contracts/MaticX.sol#325) is not in mixedCase
Parameter MaticX.migrateDrainedTokens(uint256,uint256)._idx (contracts/MaticX.sol#357) is not in mixedCase
Parameter MaticX.migrateDrainedTokens(uint256,uint256)._validatorId (contracts/MaticX.sol#357) is not in mixedCase
Parameter MaticX.buyVoucher(address,uint256,uint256)._validatorShare (contracts/MaticX.sol#396) is not in mixedCase
Parameter MaticX.buyVoucher(address,uint256,uint256)._amount (contracts/MaticX.sol#397) is not in mixedCase
Parameter MaticX.buyVoucher(address,uint256,uint256)._minSharesToMint (contracts/MaticX.sol#398) is not in mixedCase
Function MaticX.sellVoucher_new(address,uint256,uint256) (contracts/MaticX.sol#414-423) is not in mixedCase
Parameter MaticX.sellVoucher_new(address,uint256,uint256)._validatorShare (contracts/MaticX.sol#415) is not in mixedCase
Parameter MaticX.sellVoucher_new(address,uint256,uint256)._claimAmount (contracts/MaticX.sol#416) is not in mixedCase
Parameter MaticX.sellVoucher_new(address,uint256,uint256)._maximumSharesToBurn (contracts/MaticX.sol#417) is not in mixedCase
Function MaticX.unstakeClaimTokens_new(address,uint256) (contracts/MaticX.sol#430-435) is not in mixedCase
Parameter MaticX.unstakeClaimTokens_new(address,uint256)._validatorShare (contracts/MaticX.sol#431) is not in mixedCase
Parameter MaticX.unstakeClaimTokens_new(address,uint256)._unbondNonce (contracts/MaticX.sol#432) is not in mixedCase
Parameter MaticX.restake(address)._validatorShare (contracts/MaticX.sol#441) is not in mixedCase
Parameter MaticX.withdrawRewards(address)._validatorShare (contracts/MaticX.sol#452) is not in mixedCase
Parameter MaticX.getTotalStake(IValidatorShare)._validatorShare (contracts/MaticX.sol#461) is not in mixedCase
Parameter MaticX.getUserWithdrawalRequests(address)._address (contracts/MaticX.sol#555) is not in mixedCase
Parameter MaticX.getSharesAmountOfUserWithdrawalRequest(address,uint256)._address (contracts/MaticX.sol#570) is not in mixedCase
Parameter MaticX.getSharesAmountOfUserWithdrawalRequest(address,uint256)._idx (contracts/MaticX.sol#571) is not in mixedCase
Parameter MaticX.convertMaticXToMatic(uint256)._balance (contracts/MaticX.sol#590) is not in mixedCase
Parameter MaticX.convertMaticToMaticX(uint256)._balance (contracts/MaticX.sol#616) is not in mixedCase
Parameter MaticX.setFees(uint8,uint8)._treasuryFee (contracts/MaticX.sol#649) is not in mixedCase
Parameter MaticX.setFees(uint8,uint8)._insuranceFee (contracts/MaticX.sol#649) is not in mixedCase
Parameter MaticX.setFeePercent(uint8)._feePercent (contracts/MaticX.sol#662) is not in mixedCase
Parameter MaticX.setTreasuryAddress(address)._address (contracts/MaticX.sol#675) is not in mixedCase
Parameter MaticX.setInsuranceAddress(address)._address (contracts/MaticX.sol#687) is not in mixedCase
Parameter MaticX.setValidatorRegistryAddress(address)._address (contracts/MaticX.sol#700) is not in mixedCase
Parameter MaticX.setVersion(string)._version (contracts/MaticX.sol#712) is not in mixedCase
Variable MaticX.instant_pool_owner (contracts/MaticX.sol#38) is not in mixedCase
Variable MaticX.instant_pool_matic (contracts/MaticX.sol#39) is not in mixedCase
Variable MaticX.instant_pool_maticx (contracts/MaticX.sol#40) is not in mixedCase
Parameter IMaticX.initialize(address,address,address,address,address,address,address)._instant_pool_manager (contracts/interfaces/IMaticX.sol#41) is not in mixedCase
Function IValidatorShare.sellVoucher_new(uint256,uint256) (contracts/interfaces/IValidatorShare.sol#22-23) is not in mixedCase
Function IValidatorShare.unstakeClaimTokens_new(uint256) (contracts/interfaces/IValidatorShare.sol#25) is not in mixedCase
Function IValidatorShare.unbonds_new(address,uint256) (contracts/interfaces/IValidatorShare.sol#36-39) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

PausableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#102) is never used in MaticX (contracts/MaticX.sol#17-719)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

grantRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.grantRole(bytes32,address) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#136-138)
revokeRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.revokeRole(bytes32,address) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#149-151)
renounceRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.renounceRole(bytes32,address) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#167-171)
name() should be declared external:
        - ERC20Upgradeable.name() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#67-69)
symbol() should be declared external:
        - ERC20Upgradeable.symbol() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#75-77)
decimals() should be declared external:
        - ERC20Upgradeable.decimals() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#92-94)
balanceOf(address) should be declared external:
        - ERC20Upgradeable.balanceOf(address) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#106-108)
transfer(address,uint256) should be declared external:
        - ERC20Upgradeable.transfer(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#118-122)
approve(address,uint256) should be declared external:
        - ERC20Upgradeable.approve(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#141-145)
transferFrom(address,address,uint256) should be declared external:
        - ERC20Upgradeable.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#163-172)
increaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#186-190)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#206-215)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

ValidatorRegistry.sol



```
AccessControlUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#235) shadows:
        - ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#41)
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#36)
PausableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#102) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

ValidatorRegistry.initialize(address,address,address,address)._stakeManager (contracts/ValidatorRegistry.sol#42) lacks a zero-check on :
                - stakeManager = _stakeManager (contracts/ValidatorRegistry.sol#50)
ValidatorRegistry.initialize(address,address,address,address)._polygonERC20 (contracts/ValidatorRegistry.sol#43) lacks a zero-check on :
                - polygonERC20 = _polygonERC20 (contracts/ValidatorRegistry.sol#51)
ValidatorRegistry.initialize(address,address,address,address)._maticX (contracts/ValidatorRegistry.sol#44) lacks a zero-check on :
                - maticX = _maticX (contracts/ValidatorRegistry.sol#52)
ValidatorRegistry.setMaticX(address)._maticX (contracts/ValidatorRegistry.sol#157) lacks a zero-check on :
                - maticX = _maticX (contracts/ValidatorRegistry.sol#162)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#186-189)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

ValidatorRegistry.whenValidatorIdExists(uint256) (contracts/ValidatorRegistry.sol#245-251) compares to a boolean constant:
        -require(bool,string)(validatorIdExists[_validatorId] == true,Validator id doesn't exist in our registry) (contracts/ValidatorRegistry.sol#246-249)
ValidatorRegistry.whenValidatorIdDoesNotExist(uint256) (contracts/ValidatorRegistry.sol#260-266) compares to a boolean constant:
        -require(bool,string)(validatorIdExists[_validatorId] == false,Validator id already exists in our registry) (contracts/ValidatorRegistry.sol#261-264)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['0.8.7', '^0.8.0', '^0.8.1']
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/IAccessControlUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol#4)
        - 0.8.7 (contracts/ValidatorRegistry.sol#2)
        - 0.8.7 (contracts/interfaces/IMaticX.sol#2)
        - 0.8.7 (contracts/interfaces/IStakeManager.sol#2)
        - 0.8.7 (contracts/interfaces/IValidatorRegistry.sol#2)
        - 0.8.7 (contracts/interfaces/IValidatorShare.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

AccessControlUpgradeable.__AccessControl_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#54-55) is never used and should be removed
AccessControlUpgradeable._setRoleAdmin(bytes32,bytes32) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#200-204) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#85-87) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#95-101) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#114-120) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#147-149) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65) is never used and should be removed
AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) is never used and should be removed
ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is never used and should be removed
ContextUpgradeable.__Context_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is never used and should be removed
ContextUpgradeable._msgData() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#27-29) is never used and should be removed
ERC165Upgradeable.__ERC165_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#24-25) is never used and should be removed
ERC165Upgradeable.__ERC165_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#27-28) is never used and should be removed
StringsUpgradeable.toHexString(uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#40-51) is never used and should be removed
StringsUpgradeable.toString(uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#15-35) is never used and should be removed

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

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/IAccessControlUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol#4) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#137)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#164)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function AccessControlUpgradeable.__AccessControl_init() (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#51-52) is not in mixedCase
Function AccessControlUpgradeable.__AccessControl_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#54-55) is not in mixedCase
Variable AccessControlUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#235) is not in mixedCase
Function PausableUpgradeable.__Pausable_init() (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#34-36) is not in mixedCase
Function PausableUpgradeable.__Pausable_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#38-40) is not in mixedCase
```

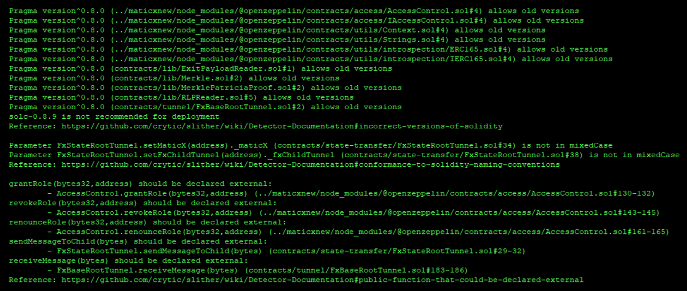
```
Variable PausableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#102) is not in mixedCase
Function ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is not in mixedCase
Variable ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#36) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#24-25) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#27-28) is not in mixedCase
Variable ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#41) is not in mixedCase
Parameter ValidatorRegistry.initialize(address,address,address,address)._stakeManager (contracts/ValidatorRegistry.sol#42) is not in mixedCase
Parameter ValidatorRegistry.initialize(address,address,address,address)._polygonERC20 (contracts/ValidatorRegistry.sol#43) is not in mixedCase
Parameter ValidatorRegistry.initialize(address,address,address,address)._maticX (contracts/ValidatorRegistry.sol#44) is not in mixedCase
Parameter ValidatorRegistry.initialize(address,address,address,address)._manager (contracts/ValidatorRegistry.sol#45) is not in mixedCase
Parameter ValidatorRegistry.addValidator(uint256)._validatorId (contracts/ValidatorRegistry.sol#62) is not in mixedCase
Parameter ValidatorRegistry.removeValidator(uint256)._validatorId (contracts/ValidatorRegistry.sol#90) is not in mixedCase
Parameter ValidatorRegistry.setPreferredDepositValidatorId(uint256)._validatorId (contracts/ValidatorRegistry.sol#130) is not in mixedCase
Parameter ValidatorRegistry.setPreferredWithdrawalValidatorId(uint256)._validatorId (contracts/ValidatorRegistry.sol#144) is not in mixedCase
Parameter ValidatorRegistry.setMaticX(address)._maticX (contracts/ValidatorRegistry.sol#157) is not in mixedCase
Parameter ValidatorRegistry.setVersion(string)._version (contracts/ValidatorRegistry.sol#167) is not in mixedCase
Parameter ValidatorRegistry.getValidatorId(uint256)._index (contracts/ValidatorRegistry.sol#205) is not in mixedCase
Parameter IMaticX.initialize(address,address,address,address,address,address,address)._instant_pool_manager (contracts/interfaces/IMaticX.sol#41) is not in mixedCase
Function IValidatorShare.sellVoucher_new(uint256,uint256) (contracts/interfaces/IValidatorShare.sol#22-23) is not in mixedCase
Function IValidatorShare.unstakeClaimTokens_new(uint256) (contracts/interfaces/IValidatorShare.sol#25) is not in mixedCase
Function IValidatorShare.unbonds_new(address,uint256) (contracts/interfaces/IValidatorShare.sol#36-39) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

AccessControlUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#235) is never used in ValidatorRegistry (contracts/ValidatorRegistry.sol#14-267)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

grantRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.grantRole(bytes32,address) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#136-138)
revokeRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.revokeRole(bytes32,address) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#149-151)
renounceRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.renounceRole(bytes32,address) (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#167-171)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

FxStateChildTunnel.sol

```
FxStateChildTunnel.setFxRootTunnel(address)._fxRootTunnel (contracts/state-transfer/FxStateChildTunnel.sol#20) lacks a zero-check on :
                - fxRootTunnel = _fxRootTunnel (contracts/state-transfer/FxStateChildTunnel.sol#25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Different versions of Solidity is used:
        - Version used: ['^0.8.0', '^0.8.7']
        - ^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/access/AccessControl.sol#4)
        - ^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/access/IAccessControl.sol#4)
        - ^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/Strings.sol#4)
        - ^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4)
        - ^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4)
        - ^0.8.7 (contracts/state-transfer/FxStateChildTunnel.sol#2)
        - ^0.8.0 (contracts/tunnel/FxBaseChildTunnel.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

AccessControl._setRoleAdmin(bytes32,bytes32) (../maticxnew/node_modules/@openzeppelin/contracts/access/AccessControl.sol#194-198) is never used and should be removed
Context._msgData() (../maticxnew/node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
FxBaseChildTunnel._processMessageFromRoot(uint256,address,bytes) (contracts/tunnel/FxBaseChildTunnel.sol#72-76) is never used and should be removed
FxBaseChildTunnel._sendMessageToRoot(bytes) (contracts/tunnel/FxBaseChildTunnel.sol#59-61) is never used and should be removed
Strings.toHexString(uint256) (../maticxnew/node_modules/@openzeppelin/contracts/utils/Strings.sol#40-51) is never used and should be removed
Strings.toString(uint256) (../maticxnew/node_modules/@openzeppelin/contracts/utils/Strings.sol#15-35) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/access/AccessControl.sol#4) allows old versions
Pragma version^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/access/IAccessControl.sol#4) allows old versions
Pragma version^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/Strings.sol#4) allows old versions
Pragma version^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4) allows old versions
Pragma version^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4) allows old versions
Pragma version^0.8.0 (contracts/tunnel/FxBaseChildTunnel.sol#2) allows old versions
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter FxStateChildTunnel.setFxRootTunnel(address)._fxRootTunnel (contracts/state-transfer/FxStateChildTunnel.sol#20) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (../maticxnew/node_modules/@openzeppelin/contracts/access/AccessControl.sol#130-132)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (../maticxnew/node_modules/@openzeppelin/contracts/access/AccessControl.sol#143-145)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (../maticxnew/node_modules/@openzeppelin/contracts/access/AccessControl.sol#161-165)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

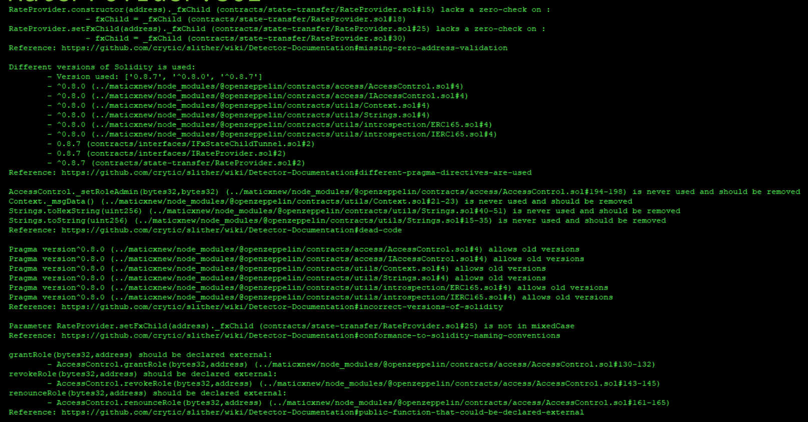
FxStateRootTunnel.sol

```
FxStateRootTunnel.constructor(address,address,address)._maticX (contracts/state-transfer/FxStateRootTunnel.sol#18) lacks a zero-check on :
                - maticX = _maticX (contracts/state-transfer/FxStateRootTunnel.sol#22)
FxStateRootTunnel.setMaticX(address)._maticX (contracts/state-transfer/FxStateRootTunnel.sol#34) lacks a zero-check on :
                - maticX = _maticX (contracts/state-transfer/FxStateRootTunnel.sol#35)
FxStateRootTunnel.setFxChildTunnel(address)._fxChildTunnel (contracts/state-transfer/FxStateRootTunnel.sol#38) lacks a zero-check on :
                - fxChildTunnel = _fxChildTunnel (contracts/state-transfer/FxStateRootTunnel.sol#43)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

ExitPayloadReader.copy(uint256,uint256,uint256) (contracts/lib/ExitPayloadReader.sol#31-55) uses assembly
        - INLINE ASM (contracts/lib/ExitPayloadReader.sol#40-42)
        - INLINE ASM (contracts/lib/ExitPayloadReader.sol#50-54)
ExitPayloadReader.getReceipt(ExitPayloadReader.ExitPayload) (contracts/lib/ExitPayloadReader.sol#115-143) uses assembly
        - INLINE ASM (contracts/lib/ExitPayloadReader.sol#132-135)
Merkle.checkMembership(bytes32,uint256,bytes32,bytes) (contracts/lib/Merkle.sol#5-37) uses assembly
        - INLINE ASM (contracts/lib/Merkle.sol#20-22)
RLPReader.toRlpItem(bytes) (contracts/lib/RLPReader.sol#52-63) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#58-60)
RLPReader.isList(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#120-131) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#125-127)
RLPReader.rlpBytesKeccak256(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#137-149) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#145-147)
RLPReader.payloadKeccak256(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#166-177) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#173-175)
RLPReader.toRlpBytes(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#182-197) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#191-193)
RLPReader.toBoolean(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#200-209) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#204-206)
RLPReader.toUint(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#218-236) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#226-233)
RLPReader.toUintStrict(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#239-250) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#245-247)
RLPReader.toBytes(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#252-266) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#260-262)
RLPReader._itemLength(uint256) (contracts/lib/RLPReader.sol#288-319) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#291-293)
        - INLINE ASM (contracts/lib/RLPReader.sol#299-305)
        - INLINE ASM (contracts/lib/RLPReader.sol#309-315)
RLPReader._payloadOffset(uint256) (contracts/lib/RLPReader.sol#322-337) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#324-326)
RLPReader.copy(uint256,uint256,uint256) (contracts/lib/RLPReader.sol#344-371) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#353-355)
        - INLINE ASM (contracts/lib/RLPReader.sol#366-370)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

FxBaseRootTunnel._validateAndExtractMessage(bytes) (contracts/tunnel/FxBaseRootTunnel.sol#72-140) compares to a boolean constant:
        -require(bool,string)(processedExits[exitHash] == false,FxRootTunnel: EXIT_ALREADY_PROCESSED) (contracts/tunnel/FxBaseRootTunnel.sol#93-96)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['^0.8.0', '^0.8.7']
        - ^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/access/AccessControl.sol#4)
        - ^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/access/IAccessControl.sol#4)
        - ^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/Strings.sol#4)
        - ^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4)
        - ^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4)
        - ^0.8.0 (contracts/lib/ExitPayloadReader.sol#1)
        - ^0.8.0 (contracts/lib/Merkle.sol#2)
        - ^0.8.0 (contracts/lib/MerklePatriciaProof.sol#2)
        - ^0.8.0 (contracts/lib/RLPReader.sol#5)
        - ^0.8.7 (contracts/state-transfer/FxStateRootTunnel.sol#2)
        - ^0.8.0 (contracts/tunnel/FxBaseRootTunnel.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

AccessControl._setRoleAdmin(bytes32,bytes32) (../maticxnew/node_modules/@openzeppelin/contracts/access/AccessControl.sol#194-198) is never used and should be removed
Context._msgData() (../maticxnew/node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
ExitPayloadReader.getBranchMaskAsUint(ExitPayloadReader.ExitPayload) (contracts/lib/ExitPayloadReader.sol#161-167) is never used and should be removed
ExitPayloadReader.toRlpBytes(ExitPayloadReader.Log) (contracts/lib/ExitPayloadReader.sol#210-212) is never used and should be removed
FxBaseRootTunnel._processMessageFromChild(bytes) (contracts/tunnel/FxBaseRootTunnel.sol#195) is never used and should be removed
RLPReader.hasNext(RLPReader.Iterator) (contracts/lib/RLPReader.sol#44-47) is never used and should be removed
RLPReader.iterator(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#70-79) is never used and should be removed
RLPReader.next(RLPReader.Iterator) (contracts/lib/RLPReader.sol#29-37) is never used and should be removed
RLPReader.payloadKeccak256(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#166-177) is never used and should be removed
RLPReader.payloadLen(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#91-93) is never used and should be removed
RLPReader.payloadLocation(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#151-160) is never used and should be removed
RLPReader.rlpBytesKeccak256(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#137-149) is never used and should be removed
RLPReader.rlpLen(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#84-86) is never used and should be removed
RLPReader.toBoolean(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#200-209) is never used and should be removed
Strings.toHexString(uint256) (../maticxnew/node_modules/@openzeppelin/contracts/utils/Strings.sol#40-51) is never used and should be removed
Strings.toString(uint256) (../maticxnew/node_modules/@openzeppelin/contracts/utils/Strings.sol#15-35) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```



```
Pragma version^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/access/AccessControl.sol#4) allows old versions
Pragma version^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/access/IAccessControl.sol#4) allows old versions
Pragma version^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/Strings.sol#4) allows old versions
Pragma version^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4) allows old versions
Pragma version^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4) allows old versions
Pragma version^0.8.0 (contracts/lib/ExitPayloadReader.sol#1) allows old versions
Pragma version^0.8.0 (contracts/lib/Merkle.sol#2) allows old versions
Pragma version^0.8.0 (contracts/lib/MerklePatriciaProof.sol#2) allows old versions
Pragma version^0.8.0 (contracts/lib/RLPReader.sol#5) allows old versions
Pragma version^0.8.0 (contracts/tunnel/FxBaseRootTunnel.sol#2) allows old versions
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter FxStateRootTunnel.setMaticX(address)._maticX (contracts/state-transfer/FxStateRootTunnel.sol#34) is not in mixedCase
Parameter FxStateRootTunnel.setFxChildTunnel(address)._fxChildTunnel (contracts/state-transfer/FxStateRootTunnel.sol#38) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (../maticxnew/node_modules/@openzeppelin/contracts/access/AccessControl.sol#130-132)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (../maticxnew/node_modules/@openzeppelin/contracts/access/AccessControl.sol#143-145)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (../maticxnew/node_modules/@openzeppelin/contracts/access/AccessControl.sol#161-165)
sendMessageToChild(bytes) should be declared external:
        - FxStateRootTunnel.sendMessageToChild(bytes) (contracts/state-transfer/FxStateRootTunnel.sol#29-32)
receiveMessage(bytes) should be declared external:
        - FxBaseRootTunnel.receiveMessage(bytes) (contracts/tunnel/FxBaseRootTunnel.sol#183-186)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

RateProvider.sol



```
RateProvider.constructor(address)._fxChild (contracts/state-transfer/RateProvider.sol#15) lacks a zero-check on :
                - fxChild = _fxChild (contracts/state-transfer/RateProvider.sol#18)
RateProvider.setFxChild(address)._fxChild (contracts/state-transfer/RateProvider.sol#25) lacks a zero-check on :
                - fxChild = _fxChild (contracts/state-transfer/RateProvider.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Different versions of Solidity is used:
        - Version used: ['0.8.7', '^0.8.0', '^0.8.7']
        - ^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/access/AccessControl.sol#4)
        - ^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/access/IAccessControl.sol#4)
        - ^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/Strings.sol#4)
        - ^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4)
        - ^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4)
        - 0.8.7 (contracts/interfaces/IFxStateChildTunnel.sol#2)
        - 0.8.7 (contracts/interfaces/IRateProvider.sol#2)
        - ^0.8.7 (contracts/state-transfer/RateProvider.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

AccessControl._setRoleAdmin(bytes32,bytes32) (../maticxnew/node_modules/@openzeppelin/contracts/access/AccessControl.sol#194-198) is never used and should be removed
Context._msgData() (../maticxnew/node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
Strings.toHexString(uint256) (../maticxnew/node_modules/@openzeppelin/contracts/utils/Strings.sol#40-51) is never used and should be removed
Strings.toString(uint256) (../maticxnew/node_modules/@openzeppelin/contracts/utils/Strings.sol#15-35) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/access/AccessControl.sol#4) allows old versions
Pragma version^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/access/IAccessControl.sol#4) allows old versions
Pragma version^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/Strings.sol#4) allows old versions
Pragma version^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4) allows old versions
Pragma version^0.8.0 (../maticxnew/node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter RateProvider.setFxChild(address)._fxChild (contracts/state-transfer/RateProvider.sol#25) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (../maticxnew/node_modules/@openzeppelin/contracts/access/AccessControl.sol#130-132)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (../maticxnew/node_modules/@openzeppelin/contracts/access/AccessControl.sol#143-145)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (../maticxnew/node_modules/@openzeppelin/contracts/access/AccessControl.sol#161-165)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

- No major issues found by Slither.
- The reentrancies flagged by Slither were checked individually and they are false positives.

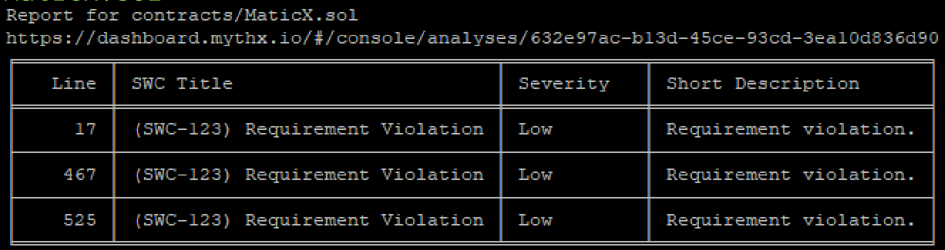
4.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on the contract and sent the compiled results to the analyzers to locate any vulnerabilities.

MythX results:

MaticX.sol



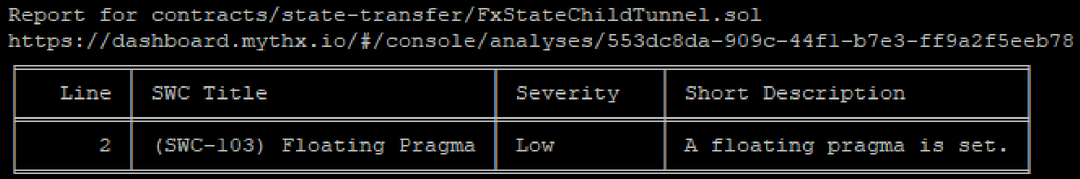
Report for contracts/MaticX.sol
https://dashboard.mythx.io/#/console/analyses/632e97ac-b13d-45ce-93cd-3ea10d836d90

Line	SWC Title	Severity	Short Description
17	(SWC-123) Requirement Violation	Low	Requirement violation.
467	(SWC-123) Requirement Violation	Low	Requirement violation.
525	(SWC-123) Requirement Violation	Low	Requirement violation.

ValidatorRegistry.sol

No issues found by MythX.

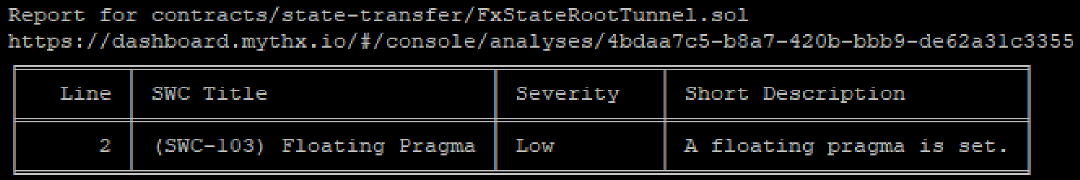
FxStateChildTunnel.sol



Report for contracts/state-transfer/FxStateChildTunnel.sol
https://dashboard.mythx.io/#/console/analyses/553dc8da-909c-44f1-b7e3-ff9a2f5eeb78

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

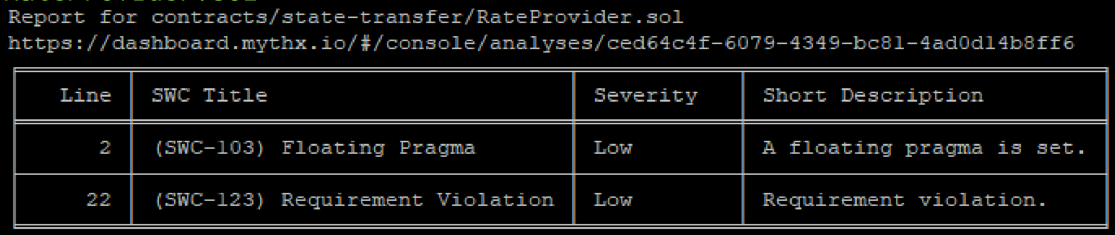
FxStateRootTunnel.sol



Report for contracts/state-transfer/FxStateRootTunnel.sol
https://dashboard.mythx.io/#/console/analyses/4bdaa7c5-b8a7-420b-bbb9-de62a31c3355

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

RateProvider.sol



```
Report for contracts/state-transfer/RateProvider.sol
https://dashboard.mythx.io/#/console/analyses/ced64c4f-6079-4349-bc81-4ad0d14b8ff6
╒════════╤══════════════════════════════════╤════════════╤═══════════════════════════╕
│   Line │ SWC Title                        │ Severity   │ Short Description         │
╞════════╪══════════════════════════════════╪════════════╪═══════════════════════════╡
│      2 │ (SWC-103) Floating Pragma        │ Low        │ A floating pragma is set. │
├────────┼──────────────────────────────────┼────────────┼───────────────────────────┤
│     22 │ (SWC-123) Requirement Violation  │ Low        │ Requirement violation.    │
╘════════╧══════════════════════════════════╧════════════╧═══════════════════════════╛
```

- No major issues found by MythX. The requirement violations are all false positives.

THANK YOU FOR CHOOSING

HALBORN