QuillAudits

Audit Report
June, 2021

Contents

Overview

Nord Finance

Repository: https://github.com/nordfinance/nordsavings-v1
Branch: Development

Scope of Audit

The scope of this audit was to analyse Nord Finance smart contract's codebase for quality, security, and correctness.

- **NordStakingWithLevels.sol:** Nord Staking Contract, which calculates and distributes bonus APY to the users according to the number of Nord tokens staked and stable tokens deposited in the vault.
  Commit: 2cb48e860039037199b236255697facb91173149

- **Vault.sol:** Vault receives deposits from users in stable tokens and mints interest bearing nTokens. On call of 'startSaving' the underlying balance is transferred to a different strategy through Fund Division Strategy. Also, it collects different types of fees such as protocol fees, treasury fees, Nord redistribution fees, stablecoin redistribution fees.
  Commit: c6ff53653e880394ffbcd16066e83c3b5132e36c

- **PolygonAaveStrategy.sol:** AaveStrategy on Polygon. Additional functions are to **claimWMatic** and **liquidateWMatic** for claiming **wmatic** from **AaveIncentiveController** and swapping it to underlying token (USDT/USDC/DAI) using **quickswap**.
  Commit: e28c3a06f6af15e62a4a7d6b49c8c5aa82496cdb

  Renamed Contract Name from **AaveStrategy** to **PolygonAaveStrategy** to avoid Duplicate Name Conflict in
  Commit: 861ff49eb58128a0d10e7da6609ff20656a66353
  This commit now represents the Fixed Version of the Contract

- **Controller.sol:** Controller to configure and interact with vaults
  Commit: d5266aa4651465e1c4715da80b599ae14b3a8cd1

- **ClaimRewardProxy.sol:** A proxy contract to getTotalRewardBalance and claim AggregatedRewards and UnstakedAmount from all three different vaults with a single call
  Commit: 5490b95976803864ac88dfbc89404470607b561b

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Malicious libraries
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- ERC20 transfer() does not return Boolean
- Using inline assembly
- ERC20 approve() race
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level calls
- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Using block.timestamp
- Multiple Sends
- Using SHA3
- Using suicide
- Using throw

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.
SmartCheck.

Static Analysis
Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to test security of smart contracts.

Code Review / Manual Analysis
Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of automated analysis were manually verified.

Gas Consumption
In this step we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Ganache, Solhint, Mythril, Slither, SmartCheck.

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity and each of them has been explained below.

High severity issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

Medium level severity issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems and they should still be fixed.

Low level severity issues

Low level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity four issues which indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Acknowledged	0	0	6	2
Closed	1	2	4	0

Issues Found – Code Review / Manual Testing

High severity issues

1. Bypassing Defense Mechanism
Vault.sol

The contract implements a defense mechanism with the help of defense() modifier with an implemented logic as: Only an EOA should be allowed to make a deposit and not a Smart Contract, and if it’s a smart contract, it should not be in the greyList. This check is implemented with the help of extcodesize opcode, which classifies a caller as an EOA or Contract depending on the code size. If it’s an EOA it returns a size of 0, and for contracts a size > 0

The defense mechanism can be bypassed by a malicious actor by calling the target function in the constructor of a contract. This way, the code size marked by extcodesize will be 0, and the contract will be considered/treated as an EOA and not a contract. This way, the actor can satisfy the first check of defense, that the caller should be an EOA, and then there won’t be a need to check for greyList.

This way, an actor can bypass both the checks.

Status: Fixed

Medium severity issues

1. **Controller.sol**

   [186-216] function **startSaving()** calls Vault's function **accumulateFees()** with incorrect order of **FeesForwarders**: Vault's function **accumulateFees()** transfers **protocol fees** to **feeForwarders[0]**, **treasuryFee** to **feeForwarders[1]**, and **buybackFee** to **feeForwarders[2]**.

   But the function **startSaving()** calls **accumulateFees()** with forwarders in the order given below

```
204         IVault(_vault).accumulateFees(
205             [protocolFeesForwarder, buybackFeesForwarder, treasuryFeesForwarder]
206         );
207         IVault(_vault).startSaving();
```

   Status: Fixed

2. **PolygonAaveStrategy.sol**

   [169-178] function **approve()** has public visibility. Approve has been used to **approve** tokens to **Aave Lending Pool** prior to the **deposit**. As the visibility is **public** and **no restrictions** for the **calls** have been implemented. It can be called by anyone, which may affect the deposit of tokens to the pool.

   Status: Fixed

Low level severity issues

1. **Older Version of the solidity compiler has been used in all the contracts: Use newer versions so as to avoid bugs introduced in the older compilers.**

   Status: Acknowledged by the Auditee

2. approve() race:

The standard ERC20 implementation contains a widely-known racing condition in its approve function, wherein a spender is able to witness the token owner broadcast a transaction altering their approval and quickly sign and broadcast a transaction using transferFrom to move the current approved amount from the owner's balance to the spender. If the spender's transaction is validated before the owner's, the spender is able to spend their entire approval amount twice.

Same issues have been found in the function safeApprove() and have been deprecated.

Reference:

- https://docs.google.com/document/d/1YLPtQxZu1UAvO9cZ1O2RPXBbT0mooh4DYKjA_jp-RLM/edit
- https://eips.ethereum.org/EIPS/eip-20

Status: Acknowledged by the Auditee

3. Incorrect Logic Implementation NordStakingWithLevels.sol

[#L586-615] function withdrawUnstakedAmountImmediate(): Current implemented logic for require check is

```
587         require(
588             !isImmediateWithdrawActive,
589             "Immediate Withdraw functionality is not active"
590         );
591
```

Meaning it will disallow immediate withdraw of unstaked amount if the boolean variable is set to true, whereas it should allow the withdrawal if set to true.

Status: Fixed

4. **NordStakingWithLevels.sol**

   [#L396, 415] block.timestamp has been used for comparisons. Avoid using block.timestamp as it can be manipulated by miners.

   Status: Acknowledged by the Auditee

5. **Vault.sol**

   [#L361, 777] block.timestamp has been used for comparisons. Avoid using block.timestamp as it can be manipulated by miners.

   Status: Acknowledged by the Auditee

6. **Vault.sol**

   A large number of **addresses** can result in **DoS**.
   [#L709-729] function **migrate()**: As the function does operations on an unknown number of **addresses**. A large number of operations may exceed the **block gas** limit and can lead to **DoS**.

   Status: Acknowledged by the Auditee

7. **Missing Zero Address Validation**
   PolygonAaveStrategy.sol

   [#L56-83] constructor(): Missing Zero Address check for _vault and _quickswap address.

   Status: Fixed

8. **Missing Already Added/Removed Checks**
   Controller.sol

   [#L93-66] function **addSaver()**, [98-101] function **removeSaver()**, [#L104-106]function **addToGreyList()** and [108-110]function **removeFromGreyList()**: Already added/removed checks can be added to add/remove a **worker** address from **savers** and a contract from the **greylist.**
   Update: GreyList Functionality has been replaced with WhiteList

   Status: Fixed

9. **Missing Zero Address Validation**
ClaimRewardProxy.sol

[#L31-47] function initializeClaimRewardProxy(): Missing Zero Address check for **rewardDistributor** and **_forwarder** address.
[#L50-60] function addRewardDistributor(): Missing Zero Address check for **_rewardDistributors** address.
[#L104-109] function setTrustedForwarder(): Missing Zero Address check for **_forwarder address**.

Status: Fixed

10. **ClaimRewardProxy.sol**

Large Number of **rewardDistributors** can result in **DoS**
[#L85-102] function **claimAggregatedRewards**() and **claimUnstakedAmount**(): As the functions do operations on an unknown number of **rewardDistributors**. A large number of operations may exceed the **block gas limit** and can lead to **DoS**.

Status: Acknowledged by the Auditee

Informational

1. **ClaimRewardProxy.sol**

[#L85-102] function claimAggregatedRewards() ignores return value by **IStakingWithLevels(rewardDistributors[i]).claimReward(_msgSender());**

Status: Acknowledged by the Auditee

2. **Ignored Returned Values**
PolygonAaveStrategy.sol

[#L241-255] function withdrawAll() ignores returned value by **lendingPool.withdraw(address(underlyingERC), amount, address(this));**

[#L273-284] function claimWMatic() ignores returned value by

```
279             IAaveIncentivesController(aaveIncentivesController).claimRewards(
280                 amTokens,
281                 rewardBalance,
282                 address(this)
283             );
```

[#L286-307] function liquidateWMatic() ignores returned value by

```
300             IUniswapV2Router02(uniswapRouterV2).swapExactTokensForTokens(
301                 balance,
302                 amountOutMin,
303                 path,
304                 address(this),
305                 block.timestamp
306             );
```

Status: Acknowledged by the Auditee

Gas Optimization

These are public functions that are never called by the contract should be declared external to save gas.

Vault.sol

```
initializeVault(address,address,uint256,uint256,address) should be declared external:
        - Vault.initializeVault(address,address,uint256,uint256,address) (Vault.sol#1669-1707)
underlyingBalanceWithInvestmentForHolder(address) should be declared external:
        - Vault.underlyingBalanceWithInvestmentForHolder(address) (Vault.sol#1894-1904)
announceStrategyUpdate(address) should be declared external:
        - Vault.announceStrategyUpdate(address) (Vault.sol#1929-1938)
setStrategy(address) should be declared external:
        - Vault.setStrategy(address) (Vault.sol#1948-1985)
accumulateFees(address[3]) should be declared external:
        - Vault.accumulateFees(address[3]) (Vault.sol#2037-2080)
withdrawAll() should be declared external:
        - Vault.withdrawAll() (Vault.sol#2123-2130)
migrate(address,address,address[]) should be declared external:
        - Vault.migrate(address,address,address[]) (Vault.sol#2270-2290)
scheduleUpgrade(address) should be declared external:
        - Vault.scheduleUpgrade(address) (Vault.sol#2328-2333)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

ClaimRewardProxy.sol

```
initializeClaimRewardProxy(address[],address) should be declared external:
        - ClaimRewardProxy.initializeClaimRewardProxy(address[],address) (ClaimRewardProxy.sol#422-438)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Controller.sol

```
getPricePerFullShare(address) should be declared external:
        - Controller.getPricePerFullShare(address) (Controller.sol#750-756)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

PolygonAaveStrategy.sol

```
underlying() should be declared external:
        - AaveStrategy.underlying() (PolygonAaveStrategy.sol#1258-1260)
investedUnderlyingBalance() should be declared external:
        - AaveStrategy.investedUnderlyingBalance() (PolygonAaveStrategy.sol#1308-1317)
startSaving() should be declared external:
        - AaveStrategy.startSaving() (PolygonAaveStrategy.sol#1338-1351)
withdrawAllToVault() should be declared external:
        - AaveStrategy.withdrawAllToVault() (PolygonAaveStrategy.sol#1376-1378)
withdrawToVault(uint256) should be declared external:
        - AaveStrategy.withdrawToVault(uint256) (PolygonAaveStrategy.sol#1385-1391)
salvage(address,address,uint256) should be declared external:
        - AaveStrategy.salvage(address,address,uint256) (PolygonAaveStrategy.sol#1415-1426)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

NordStakingWithLevels.sol

```
totalStakedFor(address) should be declared external:
        - IStakingWithLevels.totalStakedFor(address) (NordStakingWithLevels.sol#890-894)
        - NordStakingWithLevels.totalStakedFor(address) (NordStakingWithLevels.sol#1480-1487)
getRewardBalance(address) should be declared external:
        - IStakingWithLevels.getRewardBalance(address) (NordStakingWithLevels.sol#896-900)
        - NordStakingWithLevels.getRewardBalance(address) (NordStakingWithLevels.sol#1493-1500)
claimUnstakedAmount(address) should be declared external:
        - IStakingWithLevels.claimUnstakedAmount(address) (NordStakingWithLevels.sol#902)
        - NordStakingWithLevels.claimUnstakedAmount(address) (NordStakingWithLevels.sol#1680-1699)
totalUnstakedAmountReadyToClaim(address) should be declared external:
        - IStakingWithLevels.totalUnstakedAmountReadyToClaim(address) (NordStakingWithLevels.sol#904-908)
        - NordStakingWithLevels.totalUnstakedAmountReadyToClaim(address) (NordStakingWithLevels.sol#1701-1711)
exit(address) should be declared external:
        - IStakingWithLevels.exit(address) (NordStakingWithLevels.sol#910)
        - NordStakingWithLevels.exit(address) (NordStakingWithLevels.sol#1713-1725)
migrateUserStakes(address,address) should be declared external:
        - IStakingWithLevels.migrateUserStakes(address,address) (NordStakingWithLevels.sol#912-914)
        - NordStakingWithLevels.migrateUserStakes(address,address) (NordStakingWithLevels.sol#1826-1836)
migrate(IStakingWithLevels.UserData,address) should be declared external:
        - IStakingWithLevels.migrate(IStakingWithLevels.UserData,address) (NordStakingWithLevels.sol#916)
        - NordStakingWithLevels.migrate(IStakingWithLevels.UserData,address) (NordStakingWithLevels.sol#1838-1849)
withdrawUnstakedAmountImmediate(address) should be declared external:
        - IStakingWithLevels.withdrawUnstakedAmountImmediate(address) (NordStakingWithLevels.sol#925)
        - NordStakingWithLevels.withdrawUnstakedAmountImmediate(address) (NordStakingWithLevels.sol#1878-1907)
accumulateFees(address) should be declared external:
        - IStakingWithLevels.accumulateFees(address) (NordStakingWithLevels.sol#927)
        - NordStakingWithLevels.accumulateFees(address) (NordStakingWithLevels.sol#1909-1919)
setGovernance(address) should be declared external:
        - Storage.setGovernance(address) (NordStakingWithLevels.sol#944-947)
setController(address) should be declared external:
        - Storage.setController(address) (NordStakingWithLevels.sol#949-952)
isController(address) should be declared external:
        - Storage.isController(address) (NordStakingWithLevels.sol#958-960)
setStorage(address) should be declared external:
        - GovernableInit.setStorage(address) (NordStakingWithLevels.sol#1188-1191)
getUserData(address) should be declared external:
        - NordStakingWithLevels.getUserData(address) (NordStakingWithLevels.sol#1472-1478)
getUnderlyingToken() should be declared external:
        - NordStakingWithLevels.getUnderlyingToken() (NordStakingWithLevels.sol#1858-1860)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Automated Testing

Slither

Controller.sol

Event emitted after the external call

```
Reentrancy in Controller.startSaving(address,uint256,uint256,uint256) (Controller.sol#758-788):
        External calls:
        - IVault(_vault).accumulateFees((protocolFeesForwarder,buybackFeesForwarder,treasuryFeesForwarder)) (Controller.sol#776-778)
        - IVault(_vault).startSaving() (Controller.sol#779)
        Event emitted after the call(s):
        - SharePriceChangeLog(_vault,IVault(_vault).strategy(),oldSharePrice,IVault(_vault).getPricePerFullShare(),block.timestamp) (Controller.sol#781-787)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

NordStakingWithLevels.sol

State Variables Written After the External Call

```
Reentrancy in NordStakingWithLevels._unstake(address,uint256,uint256) (NordStakingWithLevels.sol#1571-1632):
        External calls:
        - _claimPartialReward(beneficiary,stableTokensToWithdraw,totalStableToken) (NordStakingWithLevels.sol#1601-1605)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (NordStakingWithLevels.sol#495)
                - (success,returndata) = target.call{value: weiValue}(data) (NordStakingWithLevels.sol#424)
                - stakeToken.safeTransfer(beneficiary,reward) (NordStakingWithLevels.sol#1768)
        - _claimReward(beneficiary) (NordStakingWithLevels.sol#1608)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (NordStakingWithLevels.sol#495)
                - stakeToken.safeTransfer(beneficiary,reward) (NordStakingWithLevels.sol#1747)
                - (success,returndata) = target.call{value: weiValue}(data) (NordStakingWithLevels.sol#424)
        - stakeToken.safeTransfer(beneficiary,stakeTokensToWithdraw) (NordStakingWithLevels.sol#1613)
        External calls sending eth:
        - _claimPartialReward(beneficiary,stableTokensToWithdraw,totalStableToken) (NordStakingWithLevels.sol#1601-1605)
                - (success,returndata) = target.call{value: weiValue}(data) (NordStakingWithLevels.sol#424)
        - _claimReward(beneficiary) (NordStakingWithLevels.sol#1608)
                - (success,returndata) = target.call{value: weiValue}(data) (NordStakingWithLevels.sol#424)
        State variables written after the call(s):
        - updateLevel(beneficiary) (NordStakingWithLevels.sol#1626)
                - users[beneficiary].level = getRewardLevel(users[beneficiary].stakeToken) (NordStakingWithLevels.sol#1639-1641)
                - users[beneficiary].rewardPercentTimes100 = level0RewardPercent (NordStakingWithLevels.sol#1668)
                - users[beneficiary].rewardPercentTimes100 = level1RewardPercent (NordStakingWithLevels.sol#1670)
                - users[beneficiary].rewardPercentTimes100 = level2RewardPercent (NordStakingWithLevels.sol#1672)
                - users[beneficiary].rewardPercentTimes100 = level3RewardPercent (NordStakingWithLevels.sol#1674)
                - users[beneficiary].rewardPercentTimes100 = level4RewardPercent (NordStakingWithLevels.sol#1676)
Reentrancy in NordStakingWithLevels.exit(address) (NordStakingWithLevels.sol#1713-1725):
        External calls:
        - _claimReward(beneficiary) (NordStakingWithLevels.sol#1719)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (NordStakingWithLevels.sol#495)
                - stakeToken.safeTransfer(beneficiary,reward) (NordStakingWithLevels.sol#1747)
                - (success,returndata) = target.call{value: weiValue}(data) (NordStakingWithLevels.sol#424)
        - _unstake(beneficiary,users[beneficiary].stakeToken,users[beneficiary].stableToken) (NordStakingWithLevels.sol#1720-1724)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (NordStakingWithLevels.sol#495)
                - stakeToken.safeTransfer(beneficiary,reward) (NordStakingWithLevels.sol#1747)
                - (success,returndata) = target.call{value: weiValue}(data) (NordStakingWithLevels.sol#424)
                - stakeToken.safeTransfer(beneficiary,reward) (NordStakingWithLevels.sol#1768)
                - stakeToken.safeTransfer(beneficiary,stakeTokensToWithdraw) (NordStakingWithLevels.sol#1613)
        External calls sending eth:
        - _claimReward(beneficiary) (NordStakingWithLevels.sol#1719)
                - (success,returndata) = target.call{value: weiValue}(data) (NordStakingWithLevels.sol#424)
        - _unstake(beneficiary,users[beneficiary].stakeToken,users[beneficiary].stableToken) (NordStakingWithLevels.sol#1720-1724)
                - (success,returndata) = target.call{value: weiValue}(data) (NordStakingWithLevels.sol#424)
        State variables written after the call(s):
        - _unstake(beneficiary,users[beneficiary].stakeToken,users[beneficiary].stableToken) (NordStakingWithLevels.sol#1720-1724)
                - users[beneficiary].level = getRewardLevel(users[beneficiary].stakeToken) (NordStakingWithLevels.sol#1639-1641)
                - users[beneficiary].rewardPercentTimes100 = level0RewardPercent (NordStakingWithLevels.sol#1668)
                - users[account].rewards = _getRewardBalance(account) (NordStakingWithLevels.sol#1400)
                - users[beneficiary].rewards = 0 (NordStakingWithLevels.sol#1746)
                - users[beneficiary].rewardPercentTimes100 = level1RewardPercent (NordStakingWithLevels.sol#1670)
                - users[beneficiary].stableToken = 0 (NordStakingWithLevels.sol#1584)
                - users[account].lastUpdateTime = lastTimeRewardApplicable() (NordStakingWithLevels.sol#1410)
                - users[beneficiary].rewardPercentTimes100 = level2RewardPercent (NordStakingWithLevels.sol#1672)
                - users[beneficiary].rewards = (users[beneficiary].rewards).sub(reward) (NordStakingWithLevels.sol#1765-1767)
                - users[beneficiary].rewardPercentTimes100 = level3RewardPercent (NordStakingWithLevels.sol#1674)
                - users[beneficiary].stakeToken = users[beneficiary].stakeToken.sub(stakeTokensToWithdraw) (NordStakingWithLevels.sol#1593-1595)
                - users[beneficiary].stableToken = users[beneficiary].stableToken.sub(stableTokensToWithdraw) (NordStakingWithLevels.sol#1596-1598)
                - users[beneficiary].rewardPercentTimes100 = level4RewardPercent (NordStakingWithLevels.sol#1676)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
```

```
Reentrancy in NordStakingWithLevels._stakeFor(address,uint256,uint256) (NordStakingWithLevels.sol#1538-1561):
        External calls:
        - stakeToken.safeTransferFrom(beneficiary,address(this),stakeTokenAmount) (NordStakingWithLevels.sol#1553-1557)
        State variables written after the call(s):
        - updateLevel(beneficiary) (NordStakingWithLevels.sol#1559)
                - users[beneficiary].level = getRewardLevel(users[beneficiary].stakeToken) (NordStakingWithLevels.sol#1639-1641)
                - users[beneficiary].rewardPercentTimes100 = level0RewardPercent (NordStakingWithLevels.sol#1668)
                - users[beneficiary].rewardPercentTimes100 = level1RewardPercent (NordStakingWithLevels.sol#1670)
                - users[beneficiary].rewardPercentTimes100 = level2RewardPercent (NordStakingWithLevels.sol#1672)
                - users[beneficiary].rewardPercentTimes100 = level3RewardPercent (NordStakingWithLevels.sol#1674)
                - users[beneficiary].rewardPercentTimes100 = level4RewardPercent (NordStakingWithLevels.sol#1676)
Reentrancy in NordStakingWithLevels.migrateUserStakes(address,address) (NordStakingWithLevels.sol#1826-1836):
        External calls:
        - IStakingWithLevels(newRewardDistributor).migrate(users[_address],_address) (NordStakingWithLevels.sol#1831-1834)
        State variables written after the call(s):
        - delete users[_address] (NordStakingWithLevels.sol#1835)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
```

```
Reentrancy in NordStakingWithLevels._unstake(address,uint256,uint256) (NordStakingWithLevels.sol#1571-1632):
        External calls:
        - _claimPartialReward(beneficiary,stableTokensToWithdraw,totalStableToken) (NordStakingWithLevels.sol#1601-1605)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (NordStakingWithLevels.sol#495)
                - (success,returndata) = target.call{value: weiValue}(data) (NordStakingWithLevels.sol#424)
                - stakeToken.safeTransfer(beneficiary,reward) (NordStakingWithLevels.sol#1768)
        - _claimReward(beneficiary) (NordStakingWithLevels.sol#1608)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (NordStakingWithLevels.sol#495)
                - stakeToken.safeTransfer(beneficiary,reward) (NordStakingWithLevels.sol#1747)
                - (success,returndata) = target.call{value: weiValue}(data) (NordStakingWithLevels.sol#424)
        External calls sending eth:
        - _claimPartialReward(beneficiary,stableTokensToWithdraw,totalStableToken) (NordStakingWithLevels.sol#1601-1605)
                - (success,returndata) = target.call{value: weiValue}(data) (NordStakingWithLevels.sol#424)
        - _claimReward(beneficiary) (NordStakingWithLevels.sol#1608)
                - (success,returndata) = target.call{value: weiValue}(data) (NordStakingWithLevels.sol#424)
        State variables written after the call(s):
        - accountUnstake.unstakingAmount = (accountUnstake.unstakingAmount).add(stakeTokensToWithdraw) (NordStakingWithLevels.sol#1618-1621)
        - accountUnstake.unstakingTime = unboundingPeriodFinish (NordStakingWithLevels.sol#1622)
Reentrancy in NordStakingWithLevels.setOldRewardDistributor(address) (NordStakingWithLevels.sol#1806-1824):
        External calls:
        - (oldRewardDistributorStakeToken,oldRewardDistributorStableToken) = IStakingWithLevels(_oldRewardDistributor).getUnderlyingToken() (NordStakingWit
hLevels.sol#1810-1813)
        State variables written after the call(s):
        - oldRewardDistributor = _oldRewardDistributor (NordStakingWithLevels.sol#1823)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

Events emitted after the external call

```
Reentrancy in NordStakingWithLevels._claimPartialReward(address,uint256,uint256) (NordStakingWithLevels.sol#1753-1772):
        External calls:
        - stakeToken.safeTransfer(beneficiary,reward) (NordStakingWithLevels.sol#1768)
        Event emitted after the call(s):
        - RewardPaid(beneficiary,reward) (NordStakingWithLevels.sol#1769)
Reentrancy in NordStakingWithLevels._claimReward(address) (NordStakingWithLevels.sol#1739-1751):
        External calls:
        - stakeToken.safeTransfer(beneficiary,reward) (NordStakingWithLevels.sol#1747)
        Event emitted after the call(s):
        - RewardPaid(beneficiary,reward) (NordStakingWithLevels.sol#1748)
Reentrancy in NordStakingWithLevels._stakeFor(address,uint256,uint256) (NordStakingWithLevels.sol#1538-1561):
        External calls:
        - stakeToken.safeTransferFrom(beneficiary,address(this),stakeTokenAmount) (NordStakingWithLevels.sol#1553-1557)
        Event emitted after the call(s):
        - Staked(beneficiary,stakeTokenAmount,stableTokenAmount) (NordStakingWithLevels.sol#1560)
Reentrancy in NordStakingWithLevels._unstake(address,uint256,uint256) (NordStakingWithLevels.sol#1571-1632):
        External calls:
        - _claimPartialReward(beneficiary,stableTokensToWithdraw,totalStableToken) (NordStakingWithLevels.sol#1601-1605)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (NordStakingWithLevels.sol#495)
                - (success,returndata) = target.call{value: weiValue}(data) (NordStakingWithLevels.sol#424)
                - stakeToken.safeTransfer(beneficiary,reward) (NordStakingWithLevels.sol#1768)
        - _claimReward(beneficiary) (NordStakingWithLevels.sol#1608)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (NordStakingWithLevels.sol#495)
                - stakeToken.safeTransfer(beneficiary,reward) (NordStakingWithLevels.sol#1747)
                - (success,returndata) = target.call{value: weiValue}(data) (NordStakingWithLevels.sol#424)
        - stakeToken.safeTransfer(beneficiary,stakeTokensToWithdraw) (NordStakingWithLevels.sol#1613)
        External calls sending eth:
        - _claimPartialReward(beneficiary,stableTokensToWithdraw,totalStableToken) (NordStakingWithLevels.sol#1601-1605)
                - (success,returndata) = target.call{value: weiValue}(data) (NordStakingWithLevels.sol#424)
        - _claimReward(beneficiary) (NordStakingWithLevels.sol#1608)
                - (success,returndata) = target.call{value: weiValue}(data) (NordStakingWithLevels.sol#424)
        Event emitted after the call(s):
        - Unstaked(beneficiary,stakeTokensToWithdraw,stableTokensToWithdraw) (NordStakingWithLevels.sol#1627-1631)
Reentrancy in NordStakingWithLevels.claimUnstakedAmount(address) (NordStakingWithLevels.sol#1680-1699):
        External calls:
        - stakeToken.safeTransfer(user,totalUnstakedAmount) (NordStakingWithLevels.sol#1697)
        Event emitted after the call(s):
        - UnstakeAmountClaimed(user,totalUnstakedAmount) (NordStakingWithLevels.sol#1698)
```

```
Reentrancy in NordStakingWithLevels.exit(address) (NordStakingWithLevels.sol#1713-1725):
        External calls:
        - _claimReward(beneficiary) (NordStakingWithLevels.sol#1719)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (NordStakingWithLevels.sol#495)
                - stakeToken.safeTransfer(beneficiary,reward) (NordStakingWithLevels.sol#1747)
                - (success,returndata) = target.call{value: weiValue}(data) (NordStakingWithLevels.sol#424)
        - _unstake(beneficiary,users[beneficiary].stakeToken,users[beneficiary].stableToken) (NordStakingWithLevels.sol#1720-1724)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (NordStakingWithLevels.sol#495)
                - stakeToken.safeTransfer(beneficiary,reward) (NordStakingWithLevels.sol#1747)
                - (success,returndata) = target.call{value: weiValue}(data) (NordStakingWithLevels.sol#424)
                - stakeToken.safeTransfer(beneficiary,reward) (NordStakingWithLevels.sol#1768)
                - stakeToken.safeTransfer(beneficiary,stakeTokensToWithdraw) (NordStakingWithLevels.sol#1613)
        External calls sending eth:
        - _claimReward(beneficiary) (NordStakingWithLevels.sol#1719)
                - (success,returndata) = target.call{value: weiValue}(data) (NordStakingWithLevels.sol#424)
        - _unstake(beneficiary,users[beneficiary].stakeToken,users[beneficiary].stableToken) (NordStakingWithLevels.sol#1720-1724)
                - (success,returndata) = target.call{value: weiValue}(data) (NordStakingWithLevels.sol#424)
        Event emitted after the call(s):
        - RewardPaid(beneficiary,reward) (NordStakingWithLevels.sol#1748)
                - _unstake(beneficiary,users[beneficiary].stakeToken,users[beneficiary].stableToken) (NordStakingWithLevels.sol#1720-1724)
        - RewardPaid(beneficiary,reward) (NordStakingWithLevels.sol#1769)
                - _unstake(beneficiary,users[beneficiary].stakeToken,users[beneficiary].stableToken) (NordStakingWithLevels.sol#1720-1724)
        - Unstaked(beneficiary,stakeTokensToWithdraw,stableTokensToWithdraw) (NordStakingWithLevels.sol#1627-1631)
                - _unstake(beneficiary,users[beneficiary].stakeToken,users[beneficiary].stableToken) (NordStakingWithLevels.sol#1720-1724)
Reentrancy in NordStakingWithLevels.recoverExcessToken(address,uint256) (NordStakingWithLevels.sol#1921-1927):
        External calls:
        - IERC20(token).safeTransfer(msg.sender,amount) (NordStakingWithLevels.sol#1925)
        Event emitted after the call(s):
        - RecoverToken(token,amount) (NordStakingWithLevels.sol#1926)
Reentrancy in NordStakingWithLevels.withdrawUnstakedAmountImmediate(address) (NordStakingWithLevels.sol#1878-1907):
        External calls:
        - stakeToken.safeTransfer(user,totalUnstakedAmount) (NordStakingWithLevels.sol#1905)
        Event emitted after the call(s):
        - UnstakeAmountClaimed(user,totalUnstakedAmount) (NordStakingWithLevels.sol#1906)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

ClaimRewardProxy.sol

External Calls Inside the loop

```
ClaimRewardProxy.claimAggregatedRewards() (ClaimRewardProxy.sol#476-493) has external calls inside a loop: IStakingWithLevels(rewardDistributors[i]).getRewardBalance(_msgSender()) > 0 (ClaimRewardProxy.sol#484-486)
ClaimRewardProxy.claimAggregatedRewards() (ClaimRewardProxy.sol#476-493) has external calls inside a loop: IStakingWithLevels(rewardDistributors[i]).claimReward(_msgSender()) (ClaimRewardProxy.sol#488-490)
ClaimRewardProxy.getTotalRewardBalance(address) (ClaimRewardProxy.sol#503-516) has external calls inside a loop: totalRewardBalance = totalRewardBalance.add(IStakingWithLevels(rewardDistributors[i]).getRewardBalance(user)) (ClaimRewardProxy.sol#511-513)
ClaimRewardProxy.claimUnstakedAmount() (ClaimRewardProxy.sol#518-530) has external calls inside a loop: IStakingWithLevels(rewardDistributors[i]).totalUnstakedAmountReadyToClaim(_msgSender()) > 0 (ClaimRewardProxy.sol#522-523)
ClaimRewardProxy.claimUnstakedAmount() (ClaimRewardProxy.sol#518-530) has external calls inside a loop: IStakingWithLevels(rewardDistributors[i]).claimUnstakedAmount(_msgSender()) (ClaimRewardProxy.sol#525-527)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```

Vault.sol

External Calls Inside the loop

```
Vault.migrate(address,address,address[]) (Vault.sol#2270-2290) has external calls inside a loop: (stakeTokenAmount,stableTokenAmount) = IStakingWithLevels(
oldRewardDistributor).totalStakedFor(_addresses[i]) (Vault.sol#2278-2281)
Vault.migrate(address,address,address[]) (Vault.sol#2270-2290) has external calls inside a loop: IStakingWithLevels(oldRewardDistributor).migrateUserStakes
(newRewardDistributor,_addresses[i]) (Vault.sol#2283-2286)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```

State Variables Written After the External Call

```
Reentrancy in Vault._deposit(uint256,uint256,address,address) (Vault.sol#2215-2263):
        External calls:
        - feeAmount = handleFees(stableTokenAmount,stakeTokenAmount.add(stakedAmount),true) (Vault.sol#2233-2238)
                - IStakingWithLevels(rewardsDistributor).getRewardLevel(stakeTokenAmount) == 4 (Vault.sol#2304-2306)
        State variables written after the call(s):
        - _mint(beneficiary,toMint) (Vault.sol#2247)
                - _totalSupply = _totalSupply.add(amount) (Vault.sol#626)
Reentrancy in Vault.accumulateFees(address[3]) (Vault.sol#2037-2080):
        External calls:
        - IERC20Upgradeable(_underlying()).safeTransfer(feeForwarders[0],protocolFees) (Vault.sol#2043-2046)
        State variables written after the call(s):
        - protocolFees = 0 (Vault.sol#2048)
Reentrancy in Vault.accumulateFees(address[3]) (Vault.sol#2037-2080):
        External calls:
        - IERC20Upgradeable(_underlying()).safeTransfer(feeForwarders[0],protocolFees) (Vault.sol#2043-2046)
        - IERC20Upgradeable(_underlying()).safeTransfer(feeForwarders[1],treasuryFee) (Vault.sol#2065-2068)
        - IERC20Upgradeable(_underlying()).safeTransfer(feeForwarders[2],buybackFee) (Vault.sol#2071-2074)
        State variables written after the call(s):
        - withdrawalFees = 0 (Vault.sol#2078)
Reentrancy in Vault.withdraw(uint256,uint256) (Vault.sol#2132-2213):
        External calls:
        - IStrategy(strategy()).withdrawAllToVault() (Vault.sol#2158)
        - IStrategy(strategy()).withdrawToVault(missingUnderlying) (Vault.sol#2168)
        - IStakingWithLevels(rewardsDistributor).unstakeFor(_msgSender(),stakeTokenAmount,underlyingAmountToWithdraw) (Vault.sol#2191-2195)
        - feeAmount = handleFees(underlyingAmountToWithdraw,stakeTokenAmount,false) (Vault.sol#2197-2198)
                - IStakingWithLevels(rewardsDistributor).getRewardLevel(stakeTokenAmount) == 4 (Vault.sol#2304-2306)
        State variables written after the call(s):
        - feeAmount = handleFees(underlyingAmountToWithdraw,stakeTokenAmount,false) (Vault.sol#2197-2198)
                - protocolFees = protocolFees.add(protocolFee) (Vault.sol#2311)
        - feeAmount = handleFees(underlyingAmountToWithdraw,stakeTokenAmount,false) (Vault.sol#2197-2198)
                - withdrawalFees = withdrawalFees.add(withdrawlFee) (Vault.sol#2319)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
```

```
Reentrancy in Vault._deposit(uint256,uint256,address,address) (Vault.sol#2215-2263):
        External calls:
        - feeAmount = handleFees(stableTokenAmount,stakeTokenAmount.add(stakedAmount),true) (Vault.sol#2233-2238)
                - IStakingWithLevels(rewardsDistributor).getRewardLevel(stakeTokenAmount) == 4 (Vault.sol#2304-2306)
        State variables written after the call(s):
        - _mint(beneficiary,toMint) (Vault.sol#2247)
                - _balances[account] = _balances[account].add(amount) (Vault.sol#627)
Reentrancy in Vault.handleFees(uint256,uint256,bool) (Vault.sol#2292-2323):
        External calls:
        - IStakingWithLevels(rewardsDistributor).getRewardLevel(stakeTokenAmount) == 4 (Vault.sol#2304-2306)
        State variables written after the call(s):
        - protocolFees = protocolFees.add(protocolFee) (Vault.sol#2311)
        - totalFees = totalFees.add(protocolFee) (Vault.sol#2312)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

Events emitted after the external call

```
Reentrancy in Vault._deposit(uint256,uint256,address,address) (Vault.sol#2215-2263):
        External calls:
        - feeAmount = handleFees(stableTokenAmount,stakeTokenAmount.add(stakedAmount),true) (Vault.sol#2233-2238)
                - IStakingWithLevels(rewardsDistributor).getRewardLevel(stakeTokenAmount) == 4 (Vault.sol#2304-2306)
        Event emitted after the call(s):
        - ProtocolFeesCollected(beneficiary,feeAmount) (Vault.sol#2240)
        - Transfer(address(0),account,amount) (Vault.sol#628)
                - _mint(beneficiary,toMint) (Vault.sol#2247)
Reentrancy in Vault._deposit(uint256,uint256,address,address) (Vault.sol#2215-2263):
        External calls:
        - feeAmount = handleFees(stableTokenAmount,stakeTokenAmount.add(stakedAmount),true) (Vault.sol#2233-2238)
                - IStakingWithLevels(rewardsDistributor).getRewardLevel(stakeTokenAmount) == 4 (Vault.sol#2304-2306)
        - IStakingWithLevels(rewardsDistributor).stakeFor(beneficiary,stakeTokenAmount,depositStableTokenAmount) (Vault.sol#2249-2253)
        - IERC20Upgradeable(underlying()).safeTransferFrom(sender,address(this),stableTokenAmount) (Vault.sol#2255-2259)
        Event emitted after the call(s):
        - Deposit(beneficiary,stakeTokenAmount,stableTokenAmount) (Vault.sol#2262)
Reentrancy in Vault.accumulateFees(address[3]) (Vault.sol#2037-2080):
        External calls:
        - IERC20Upgradeable(_underlying()).safeTransfer(feeForwarders[0],protocolFees) (Vault.sol#2043-2046)
        Event emitted after the call(s):
        - ProtocolFeesTransferred(protocolFees) (Vault.sol#2047)
Reentrancy in Vault.accumulateFees(address[3]) (Vault.sol#2037-2080):
        External calls:
        - IERC20Upgradeable(_underlying()).safeTransfer(feeForwarders[0],protocolFees) (Vault.sol#2043-2046)
        - IERC20Upgradeable(_underlying()).safeTransfer(feeForwarders[1],treasuryFee) (Vault.sol#2065-2068)
        Event emitted after the call(s):
        - TreasuryFeesTransferred(treasuryFee) (Vault.sol#2069)
Reentrancy in Vault.accumulateFees(address[3]) (Vault.sol#2037-2080):
        External calls:
        - IERC20Upgradeable(_underlying()).safeTransfer(feeForwarders[0],protocolFees) (Vault.sol#2043-2046)
        - IERC20Upgradeable(_underlying()).safeTransfer(feeForwarders[1],treasuryFee) (Vault.sol#2065-2068)
        - IERC20Upgradeable(_underlying()).safeTransfer(feeForwarders[2],buybackFee) (Vault.sol#2071-2074)
        Event emitted after the call(s):
        - BuybackFeesTransferred(buybackFee) (Vault.sol#2075)
        - TotalFeesRedistributed(redistributionAmount) (Vault.sol#2077)
```

```
Reentrancy in Vault.invest() (Vault.sol#2082-2091):
        External calls:
        - IERC20Upgradeable(underlying()).safeTransfer(address(strategy()),availableAmount) (Vault.sol#2085-2088)
        Event emitted after the call(s):
        - Invest(availableAmount) (Vault.sol#2089)
Reentrancy in Vault.startSaving() (Vault.sol#1811-1834):
        External calls:
        - IStrategy(strategy()).withdrawAllToVault() (Vault.sol#1818)
        - invest() (Vault.sol#1822)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (Vault.sol#359)
                - IERC20Upgradeable(underlying()).safeTransfer(address(strategy()),availableAmount) (Vault.sol#2085-2088)
                - (success,returndata) = target.call{value: value}(data) (Vault.sol#981)
        External calls sending eth:
        - invest() (Vault.sol#1822)
                - (success,returndata) = target.call{value: value}(data) (Vault.sol#981)
        Event emitted after the call(s):
        - Invest(availableAmount) (Vault.sol#2089)
                - invest() (Vault.sol#1822)
Reentrancy in Vault.withdraw(uint256,uint256) (Vault.sol#2132-2213):
        External calls:
        - IStrategy(strategy()).withdrawAllToVault() (Vault.sol#2158)
        - IStrategy(strategy()).withdrawToVault(missingUnderlying) (Vault.sol#2168)
        - IStakingWithLevels(rewardsDistributor).unstakeFor(_msgSender(),stakeTokenAmount,underlyingAmountToWithdraw) (Vault.sol#2191-2195)
        - feeAmount = handleFees(underlyingAmountToWithdraw,stakeTokenAmount,false) (Vault.sol#2197-2198)
                - IStakingWithLevels(rewardsDistributor).getRewardLevel(stakeTokenAmount) == 4 (Vault.sol#2304-2306)
        Event emitted after the call(s):
        - WithdrawalFeesCollected(msg.sender,feeAmount) (Vault.sol#2199)
Reentrancy in Vault.withdraw(uint256,uint256) (Vault.sol#2132-2213):
        External calls:
        - IStrategy(strategy()).withdrawAllToVault() (Vault.sol#2158)
        - IStrategy(strategy()).withdrawToVault(missingUnderlying) (Vault.sol#2168)
        - IStakingWithLevels(rewardsDistributor).unstakeFor(_msgSender(),stakeTokenAmount,underlyingAmountToWithdraw) (Vault.sol#2191-2195)
        - feeAmount = handleFees(underlyingAmountToWithdraw,stakeTokenAmount,false) (Vault.sol#2197-2198)
                - IStakingWithLevels(rewardsDistributor).getRewardLevel(stakeTokenAmount) == 4 (Vault.sol#2304-2306)
        - IERC20Upgradeable(underlying()).safeTransfer(_msgSender(),underlyingAmountToWithdraw) (Vault.sol#2202-2205)
        Event emitted after the call(s):
        - Withdraw(_msgSender(),stakeTokenAmount,underlyingAmountToWithdraw) (Vault.sol#2208-2212)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

PolygonAaveStrategy.sol

Events emitted after the external call

```
Reentrancy in AaveStrategy.startSaving() (PolygonAaveStrategy.sol#1338-1351):
        External calls:
        - claimWMatic() (PolygonAaveStrategy.sol#1340)
                - IAaveIncentivesController(aaveIncentivesController).claimRewards(amTokens,rewardBalance,address(this)) (PolygonAaveStrategy.sol#1434-1438)
        - liquidateWMatic() (PolygonAaveStrategy.sol#1341)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (PolygonAaveStrategy.sol#715)
                - (success,returndata) = target.call{value: weiValue}(data) (PolygonAaveStrategy.sol#287)
                - IERC20(address(WMatic)).safeApprove(address(uniswapRouterV2),0) (PolygonAaveStrategy.sol#1450)
                - IERC20(address(WMatic)).safeApprove(address(uniswapRouterV2),balance) (PolygonAaveStrategy.sol#1451)
                - IUniswapV2Router02(uniswapRouterV2).swapExactTokensForTokens(balance,amountOutMin,path,address(this),block.timestamp) (PolygonAaveStrategy.sol#1455-1461)
        External calls sending eth:
        - liquidateWMatic() (PolygonAaveStrategy.sol#1341)
                - (success,returndata) = target.call{value: weiValue}(data) (PolygonAaveStrategy.sol#287)
        Event emitted after the call(s):
        - TooLowBalance() (PolygonAaveStrategy.sol#1444)
                - liquidateWMatic() (PolygonAaveStrategy.sol#1341)
Reentrancy in AaveStrategy.withdrawAll(uint256,address) (PolygonAaveStrategy.sol#1396-1410):
        External calls:
        - claimWMatic() (PolygonAaveStrategy.sol#1398)
                - IAaveIncentivesController(aaveIncentivesController).claimRewards(amTokens,rewardBalance,address(this)) (PolygonAaveStrategy.sol#1434-1438)
        - liquidateWMatic() (PolygonAaveStrategy.sol#1399)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (PolygonAaveStrategy.sol#715)
                - (success,returndata) = target.call{value: weiValue}(data) (PolygonAaveStrategy.sol#287)
                - IERC20(address(WMatic)).safeApprove(address(uniswapRouterV2),0) (PolygonAaveStrategy.sol#1450)
                - IERC20(address(WMatic)).safeApprove(address(uniswapRouterV2),balance) (PolygonAaveStrategy.sol#1451)
                - IUniswapV2Router02(uniswapRouterV2).swapExactTokensForTokens(balance,amountOutMin,path,address(this),block.timestamp) (PolygonAaveStrategy.sol#1455-1461)
        External calls sending eth:
        - liquidateWMatic() (PolygonAaveStrategy.sol#1399)
                - (success,returndata) = target.call{value: weiValue}(data) (PolygonAaveStrategy.sol#287)
        Event emitted after the call(s):
        - TooLowBalance() (PolygonAaveStrategy.sol#1444)
                - liquidateWMatic() (PolygonAaveStrategy.sol#1399)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

Mythril

Mythril didn’t detect any high severity issues.

Smartcheck

Smartcheck didn’t detect any high severity issues.

Solhint

Solhint didn’t detect any high severity issues.

Disclaimer

The audit does not give any warranties on the security of the code. One audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of the code. Besides a security audit, please don't consider this report as investment advice.

Closing Summary

Some issues of medium and low severity have been reported during the audit. A high issue has been reported regarding the use of **extcodesize**. Some suggestions have also been made to improve the code quality and gas optimisation.
However, most of them have now been fixed and reviewed.

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com