CYBER UNIT
Cyber Security Strategic Partner

www.cyberunit.tech

Impermax Periphery Smart Contract Audit

Date: February 24, 2021
Report for: Impermax Finance
By: CyberUnit.Tech

This document may contain confidential information about IT systems and the customer's intellectual property and information about potential vulnerabilities and exploitation methods.
The report contains confidential information. This information can be used internally by the customer. The customer can release the information after fixing all vulnerabilities.

Document

Name	Impermax Periphery
Platform	EVM
Name	Impermax-x-uniswapv2-router
Date	Feb 24, 2021

Table of contents

Introduction

This report presents the Customer` s smart contract's security assessment findings and its code review conducted between January 18 – February 3 2021.

Scope

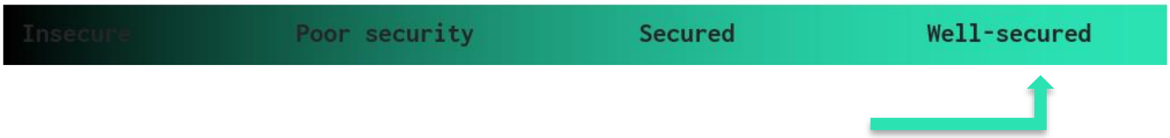
The scope of the project is Impermax-x-uniswapv2-router smart contract.

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the widely known vulnerabilities that are considered (the full list includes them but does not limit by them):

- Reentrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Compiler version not fixed
- Unchecked external call - Unchecked math
- Unsafe type inference
- Implicit visibility level

Executive Summary

According to the assessment, Customer' smart contracts are well-secured.



Our team performed an analysis of code functionality, manual audit, and automated checks with Slither and remix IDE (see Appendix B pic 1). All issues found during the automated investigation reviewed have been manually, and application vulnerabilities are presented in the Audit overview section. A general overview is shown in the AS-IS section, and all found issues can be found in the Audit overview section.

We found one low and didn't find any medium, high, and critical issues in a smart contract.

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also significantly impact smart contract execution, e.g., public access to crucial functions.
Medium	Medium-level vulnerabilities are essential to fix; however, they can't lead to tokens loss.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc., code snippets that can't significantly impact execution.
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations, and info statements can't affect smart contract execution and can be ignored.

AS-IS overview

Impermax x Uniswap V2 Router contract consists of the next smart contracts:

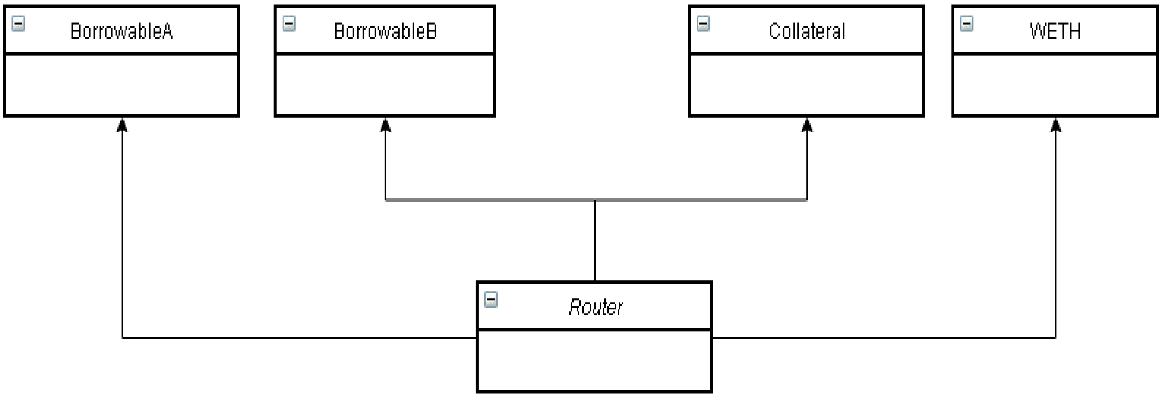
1. **SafeMath.sol,UQ112x112.sol,TransferHelper.sol, UniswapV2Library.sol** contracts – supporting libraries
2. **IBorrowable.sol, ICollateral.sol, IERC20.sol, IImpermaxCallee.sol, IPoolToken.sol, IRouter01.sol, IUniswapV2Pair.sol, IWETH.sol** contracts – interfaces
3. **Impermax** contract – **Router01.sol**

Contracts from point one were compared to original "Openzeppelin," and "Uniswap-v2-core" templates; no logic differences were found. They are considered secure.

Contracts from point 2 Impermax Interfaces - describe the actions that an object can perform.

Contracts from point 3 The Impermax classes implementing the "Impermax x Uniswap V2" protocol will be detailed in the report.

Router01 :



Router01 contract inherits the IRouter01,IImpermaxCallee

PrestakingProvisioner contract **init** function was called with following parameters:

- address(_factory)
- address(_bDeployer)
- address(_cDeployer)
- address(_WETH)

mint function was called with following parameters:

- address(poolToken)
- address(underlying)
- uint(amount)
- address(to)

_mint function was called with following parameters:

- address(poolToken)
- address(underlying)
- uint(amount)
- address(from)
- address(to)

mintETH function was called with following parameters:

- address(poolToken)
- address(to)
- uint(deadline)

mintCollateral function was called with following parameters:

- address(poolToken)
- uint(amount)
- address(to)
- uint(deadline)
- bytes calldata (permitData)

redeem function was called with following parameters:

- address(poolToken)
- uint(tokens)
- address(to)
- uint(deadline)
- bytes calldata (permitData)

borrow function was called with following parameters:

- address(borrowable)
- uint(amount)
- address(to)
- uint(deadline)
- bytes calldata (permitData)

borrowETH function was called with following parameters:

- address(borrowable)
- uint(amountETH)
- address(to)
- uint(deadline)
- bytes calldata (permitData)

_repayAmount function was called with following parameters:

- address(borrowable)
- uint(amountMax)
- address(borrower)

repay function was called with following parameters:

- address(borrowable)
- uint(amountMax)
- address(borrower)
- uint(deadline)

repayETH function was called with following parameters:

- address(borrowable)
- address(borrower)
- uint(deadline)

liquidate function was called with following parameters:

- address(borrowable)
- uint(amountMax)
- address(borrower)
- address(to)
- uint(deadline)

liquidateETH function was called with following parameters:

- address(borrowable)
- address(borrower)
- address(to)
- uint(deadline)

_leverage function was called with following parameters:

- address(uniswapV2Pair)
- uint(amountA)

- uint(amountB)
- address(to)

leverage function was called with following parameters:

- address(uniswapV2Pair)
- uint(amountADesired)
- uint(amountBDesired)
- uint(amountAMin)
- uint(amountBMin)
- address(to)
- uint(deadline)
- bytes calldata (permitDataA)
- bytes calldata (permitDataB)

_addLiquidityAndMint function was called with following

parameters:

- address(uniswapV2Pair)
- uint(amountAMin)
- uint(amountBMin)
- address(to)

impermaxBorrow function was called with followingт parameters:

- address(sender)
- address(borrower)
- uint(borrowAmount)
- bytes calldata (data)

impermaxRedeem function was called with following parameters:

- address(sender)
- uint(redeemAmount)
- bytes calldata (data)

_permit function was called with following parameters:

- address(poolToken)
- uint(amount)
- uint(deadline)
- bytes memory(permitData)

_permitUniswapV2Pair function was called with following parameters:

- address(uniswapV2Pair)
- uint(amount)
- uint(deadline)
- bytes memory(permitData)

_borrowPermit function was called with following parameters:

- address(borrowable)
- uint(amount)
- uint(deadline)
- bytes memory(permitData)

getBorrowable function was called with the following parameters:

- address(uniswapV2Pair)
- uint8(index)

getCollateral function was called with the following parameters:

- address(uniswapV2Pair)

getLendingPool function was called with the following parameters:

- address(uniswapV2Pair)

Audit overview

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

Different versions of Solidity are used in Version used: ['=0.6.6', '>=0.5.0'] (see Appendix A pic. 1 for evidence)

Conclusion Router01

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, a high-level description of functionality was presented in the report's As-is overview section. Please note that for automatic testing of the Router01 contract, we replaced the hash of the sum (see Appendix A pic. 2 for evidence).

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

The overall quality of the reviewed contracts is well-secured. Security engineers found one low vulnerability, which couldn't have any significant security impact.

Disclaimers

Disclaimer

The smart contracts given for audit had been analyzed following the best industry practices at the date of this report, concerning: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

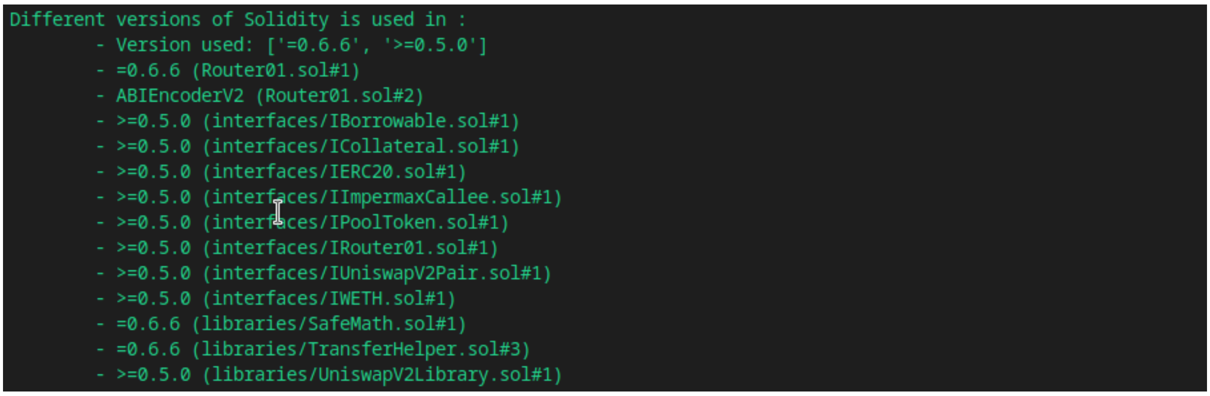
The audit makes no statements or warranties on the security of the code. It can also not be considered a sufficient assessment regarding the code's utility and safety, bug-free status, or any other contract statements. While we have done our best to conduct the analysis and produce this report, it is essential to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, programming language, and other software related to the smart contract can have their vulnerabilities leading to hacks. Thus, the audit can’t guarantee the explicit security of the audited smart contracts.

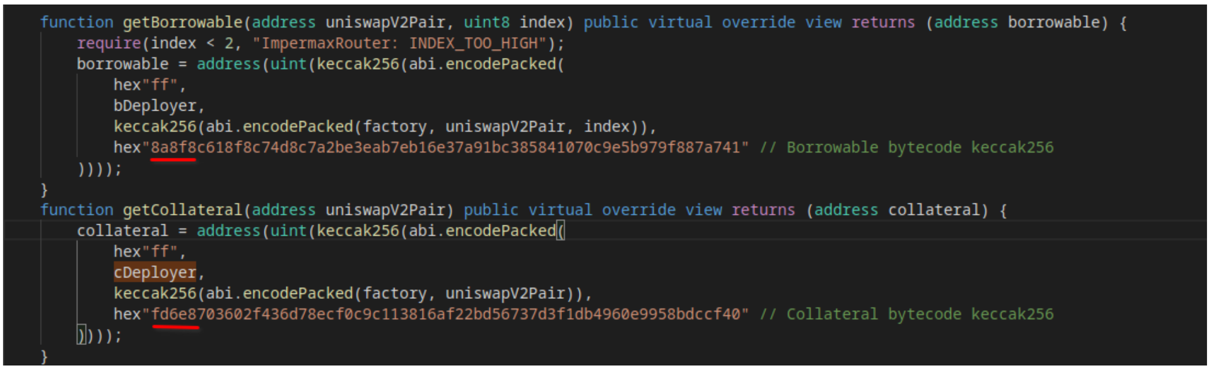
Appendix A. Evidences

Pic 1. Different pragma directives are used:



```
Different versions of Solidity is used in :
        - Version used: ['=0.6.6', '>=0.5.0']
        - =0.6.6 (Router01.sol#1)
        - ABIEncoderV2 (Router01.sol#2)
        - >=0.5.0 (interfaces/IBorrowable.sol#1)
        - >=0.5.0 (interfaces/ICollateral.sol#1)
        - >=0.5.0 (interfaces/IERC20.sol#1)
        - >=0.5.0 (interfaces/IImpermaxCallee.sol#1)
        - >=0.5.0 (interfaces/IPoolToken.sol#1)
        - >=0.5.0 (interfaces/IRouter01.sol#1)
        - >=0.5.0 (interfaces/IUniswapV2Pair.sol#1)
        - >=0.5.0 (interfaces/IWETH.sol#1)
        - =0.6.6 (libraries/SafeMath.sol#1)
        - =0.6.6 (libraries/TransferHelper.sol#3)
        - >=0.5.0 (libraries/UniswapV2Library.sol#1)
```

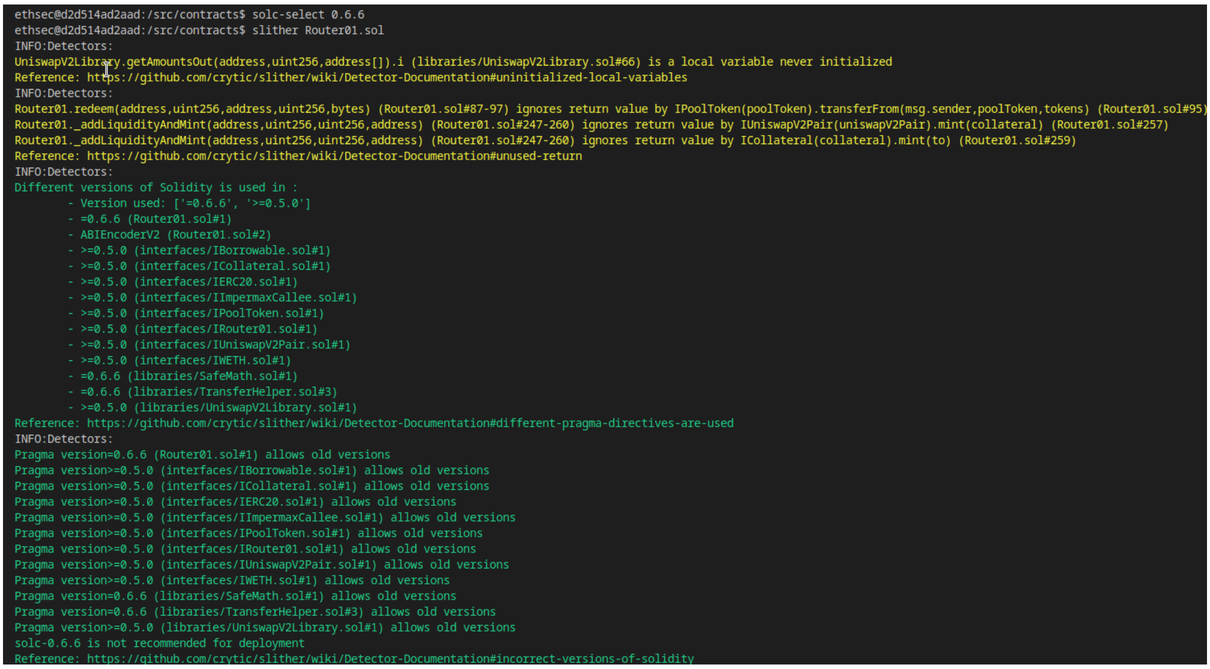
Pic 2. Different pragma directives are used:



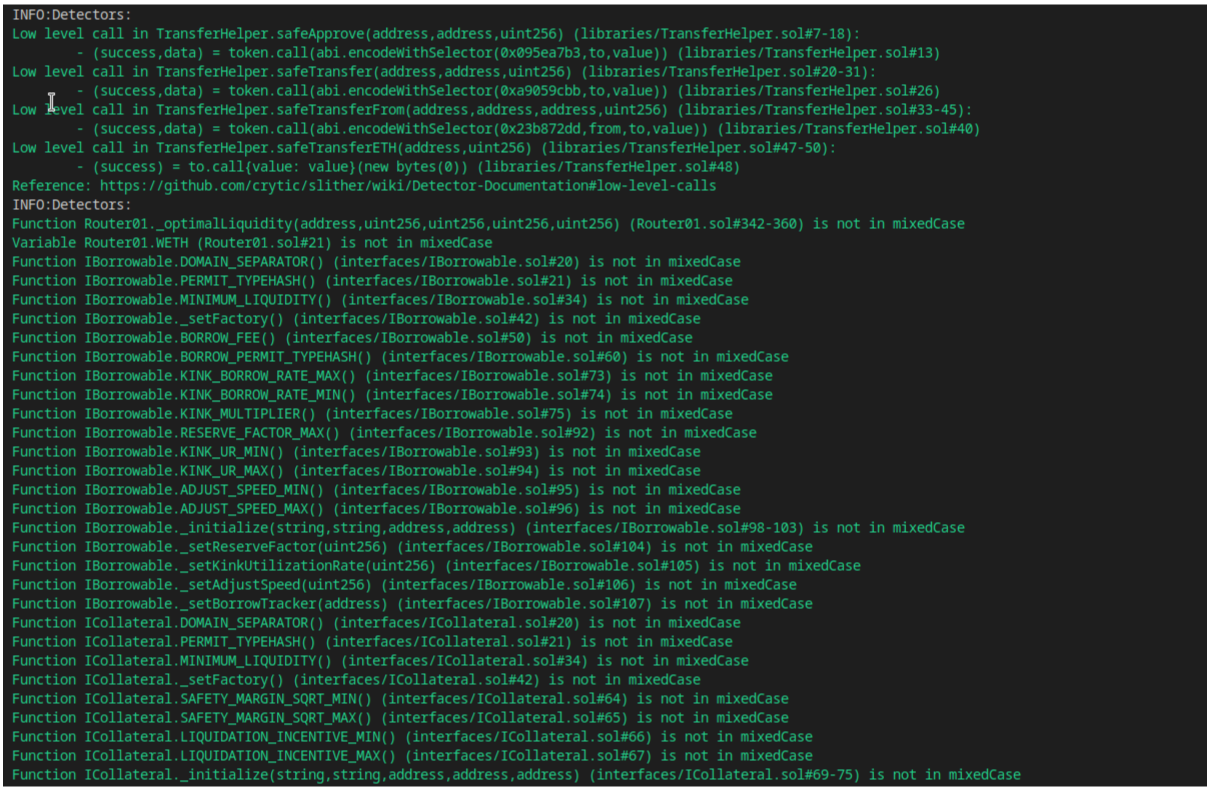
```
function getBorrowable(address uniswapV2Pair, uint8 index) public virtual override view returns (address borrowable) {
    require(index < 2, "ImpermaxRouter: INDEX_TOO_HIGH");
    borrowable = address(uint(keccak256(abi.encodePacked(
        hex"ff",
        bDeployer,
        keccak256(abi.encodePacked(factory, uniswapV2Pair, index)),
        hex"8a8f8c618f8c74d8c7a2be3eab7eb16e37a91bc385841070c9e5b979f887a741" // Borrowable bytecode keccak256
    ))));
}
function getCollateral(address uniswapV2Pair) public virtual override view returns (address collateral) {
    collateral = address(uint(keccak256(abi.encodePacked(
        hex"ff",
        cDeployer,
        keccak256(abi.encodePacked(factory, uniswapV2Pair)),
        hex"fd6e8703602f436d78ecf0c9c113816af22bd56737d3f1db4960e9958bdccf40" // Collateral bytecode keccak256
    ))));
}
```

Appendix B. Automated tools reports

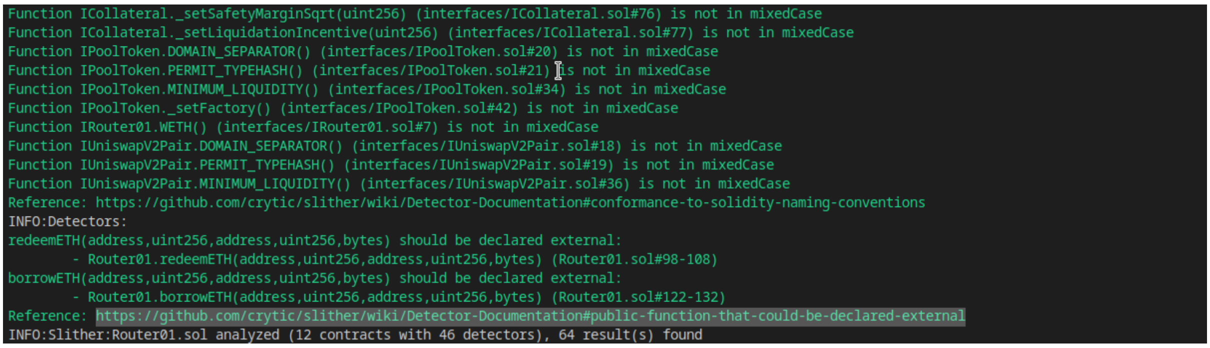
Pic 1. BAllowance Slither automated report:



```
ethsec@d2d514ad2aad:/src/contracts$ solc-select 0.6.6
ethsec@d2d514ad2aad:/src/contracts$ slither Router01.sol
INFO:Detectors:
UniswapV2Library.getAmountsOut(address,uint256,address[]).i (libraries/UniswapV2Library.sol#66) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
Router01.redeem(address,uint256,address,uint256,bytes) (Router01.sol#87-97) ignores return value by IPoolToken(poolToken).transferFrom(msg.sender,poolToken,tokens) (Router01.sol#95)
Router01._addLiquidityAndMint(address,uint256,uint256,address) (Router01.sol#247-260) ignores return value by IUniswapV2Pair(uniswapV2Pair).mint(collateral) (Router01.sol#257)
Router01._addLiquidityAndMint(address,uint256,uint256,address) (Router01.sol#247-260) ignores return value by ICollateral(collateral).mint(to) (Router01.sol#259)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Different versions of Solidity is used in :
        - Version used: ['=0.6.6', '>=0.5.0']
        - =0.6.6 (Router01.sol#1)
        - ABIEncoderV2 (Router01.sol#2)
        - >=0.5.0 (interfaces/IBorrowable.sol#1)
        - >=0.5.0 (interfaces/ICollateral.sol#1)
        - >=0.5.0 (interfaces/IERC20.sol#1)
        - >=0.5.0 (interfaces/IImpermaxCallee.sol#1)
        - >=0.5.0 (interfaces/IPoolToken.sol#1)
        - >=0.5.0 (interfaces/IRouter01.sol#1)
        - >=0.5.0 (interfaces/IUniswapV2Pair.sol#1)
        - >=0.5.0 (interfaces/IWETH.sol#1)
        - =0.6.6 (libraries/SafeMath.sol#1)
        - =0.6.6 (libraries/TransferHelper.sol#3)
        - >=0.5.0 (libraries/UniswapV2Library.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Pragma version=0.6.6 (Router01.sol#1) allows old versions
Pragma version>=0.5.0 (interfaces/IBorrowable.sol#1) allows old versions
Pragma version>=0.5.0 (interfaces/ICollateral.sol#1) allows old versions
Pragma version>=0.5.0 (interfaces/IERC20.sol#1) allows old versions
Pragma version>=0.5.0 (interfaces/IImpermaxCallee.sol#1) allows old versions
Pragma version>=0.5.0 (interfaces/IPoolToken.sol#1) allows old versions
Pragma version>=0.5.0 (interfaces/IRouter01.sol#1) allows old versions
Pragma version>=0.5.0 (interfaces/IUniswapV2Pair.sol#1) allows old versions
Pragma version>=0.5.0 (interfaces/IWETH.sol#1) allows old versions
Pragma version=0.6.6 (libraries/SafeMath.sol#1) allows old versions
Pragma version=0.6.6 (libraries/TransferHelper.sol#3) allows old versions
Pragma version>=0.5.0 (libraries/UniswapV2Library.sol#1) allows old versions
solc-0.6.6 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```



```
INFO:Detectors:
Low level call in TransferHelper.safeApprove(address,address,uint256) (libraries/TransferHelper.sol#7-18):
        - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (libraries/TransferHelper.sol#13)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (libraries/TransferHelper.sol#20-31):
        - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (libraries/TransferHelper.sol#26)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (libraries/TransferHelper.sol#33-45):
        - (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (libraries/TransferHelper.sol#40)
Low level call in TransferHelper.safeTransferETH(address,uint256) (libraries/TransferHelper.sol#47-50):
        - (success) = to.call{value: value}(new bytes(0)) (libraries/TransferHelper.sol#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function Router01._optimalLiquidity(address,uint256,uint256,uint256,uint256) (Router01.sol#342-360) is not in mixedCase
Variable Router01.WETH (Router01.sol#21) is not in mixedCase
Function IBorrowable.DOMAIN_SEPARATOR() (interfaces/IBorrowable.sol#20) is not in mixedCase
Function IBorrowable.PERMIT_TYPEHASH() (interfaces/IBorrowable.sol#21) is not in mixedCase
Function IBorrowable.MINIMUM_LIQUIDITY() (interfaces/IBorrowable.sol#34) is not in mixedCase
Function IBorrowable._setFactory() (interfaces/IBorrowable.sol#42) is not in mixedCase
Function IBorrowable.BORROW_FEE() (interfaces/IBorrowable.sol#50) is not in mixedCase
Function IBorrowable.BORROW_PERMIT_TYPEHASH() (interfaces/IBorrowable.sol#60) is not in mixedCase
Function IBorrowable.KINK_BORROW_RATE_MAX() (interfaces/IBorrowable.sol#73) is not in mixedCase
Function IBorrowable.KINK_BORROW_RATE_MIN() (interfaces/IBorrowable.sol#74) is not in mixedCase
Function IBorrowable.KINK_MULTIPLIER() (interfaces/IBorrowable.sol#75) is not in mixedCase
Function IBorrowable.RESERVE_FACTOR_MAX() (interfaces/IBorrowable.sol#92) is not in mixedCase
Function IBorrowable.KINK_UR_MIN() (interfaces/IBorrowable.sol#93) is not in mixedCase
Function IBorrowable.KINK_UR_MAX() (interfaces/IBorrowable.sol#94) is not in mixedCase
Function IBorrowable.ADJUST_SPEED_MIN() (interfaces/IBorrowable.sol#95) is not in mixedCase
Function IBorrowable.ADJUST_SPEED_MAX() (interfaces/IBorrowable.sol#96) is not in mixedCase
Function IBorrowable._initialize(string,string,address,address) (interfaces/IBorrowable.sol#98-103) is not in mixedCase
Function IBorrowable._setReserveFactor(uint256) (interfaces/IBorrowable.sol#104) is not in mixedCase
Function IBorrowable._setKinkUtilizationRate(uint256) (interfaces/IBorrowable.sol#105) is not in mixedCase
Function IBorrowable._setAdjustSpeed(uint256) (interfaces/IBorrowable.sol#106) is not in mixedCase
Function IBorrowable._setBorrowTracker(address) (interfaces/IBorrowable.sol#107) is not in mixedCase
Function ICollateral.DOMAIN_SEPARATOR() (interfaces/ICollateral.sol#20) is not in mixedCase
Function ICollateral.PERMIT_TYPEHASH() (interfaces/ICollateral.sol#21) is not in mixedCase
Function ICollateral.MINIMUM_LIQUIDITY() (interfaces/ICollateral.sol#34) is not in mixedCase
Function ICollateral._setFactory() (interfaces/ICollateral.sol#42) is not in mixedCase
Function ICollateral.SAFETY_MARGIN_SQRT_MIN() (interfaces/ICollateral.sol#64) is not in mixedCase
Function ICollateral.SAFETY_MARGIN_SQRT_MAX() (interfaces/ICollateral.sol#65) is not in mixedCase
Function ICollateral.LIQUIDATION_INCENTIVE_MIN() (interfaces/ICollateral.sol#66) is not in mixedCase
Function ICollateral.LIQUIDATION_INCENTIVE_MAX() (interfaces/ICollateral.sol#67) is not in mixedCase
Function ICollateral._initialize(string,string,address,address,address) (interfaces/ICollateral.sol#69-75) is not in mixedCase
```



```
Function ICollateral._setSafetyMarginSqrt(uint256) (interfaces/ICollateral.sol#76) is not in mixedCase
Function ICollateral._setLiquidationIncentive(uint256) (interfaces/ICollateral.sol#77) is not in mixedCase
Function IPoolToken.DOMAIN_SEPARATOR() (interfaces/IPoolToken.sol#20) is not in mixedCase
Function IPoolToken.PERMIT_TYPEHASH() (interfaces/IPoolToken.sol#21) is not in mixedCase
Function IPoolToken.MINIMUM_LIQUIDITY() (interfaces/IPoolToken.sol#34) is not in mixedCase
Function IPoolToken._setFactory() (interfaces/IPoolToken.sol#42) is not in mixedCase
Function IRouter01.WETH() (interfaces/IRouter01.sol#7) is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (interfaces/IUniswapV2Pair.sol#18) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (interfaces/IUniswapV2Pair.sol#19) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (interfaces/IUniswapV2Pair.sol#36) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
redeemETH(address,uint256,address,uint256,bytes) should be declared external:
        - Router01.redeemETH(address,uint256,address,uint256,bytes) (Router01.sol#98-108)
borrowETH(address,uint256,address,uint256,bytes) should be declared external:
        - Router01.borrowETH(address,uint256,address,uint256,bytes) (Router01.sol#122-132)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Router01.sol analyzed (12 contracts with 46 detectors), 64 result(s) found
```