QuillAudits

Audit Report
January, 2022

For

UniFarm
by OroPocket

Contents

Scope of the Audit 01

Check Vulnerabilities 01

Techniques and Methods 02

Issue Categories 03

Number of security issues per severity. 03

Introduction 04

A. Contract – GovernorBravoDelegate 05

Issues Found – Code Review / Manual Testing 05

High Severity Issues 05

Medium Severity Issues 05

Low Severity Issues 05

A.1 Missing Address Verification: 05

A.2 Missing Value Verification: 06

A.3 Floating Pragma: 07

Functional Test 08

B. Contract – GovernorBravoDelegator 09

Issues Found – Code Review / Manual Testing 09

High Severity Issues 09

Contents

Medium Severity Issues 09

Low Severity Issues 09

Functional Test 10

C. Contract – AMMUtility 11

Issues Found – Code Review / Manual Testing 11

High Severity Issues 11

Medium Severity Issues 11

Low Severity Issues 11

C.1 Missing Value Verification: 11

Functional Test 12

D. Contract – Unifarm Factory 13

Issues Found – Code Review / Manual Testing 13

High Severity Issues 13

Medium Severity Issues 13

Low Severity Issues 13

Functional Test 14

E. Contract – UnifarmPair 15

Issues Found – Code Review / Manual Testing 15

Contents

High Severity Issues 15

Medium Severity Issues 15

Low Severity Issues 15

Functional Test 16

F. Contract – UnifarmRouter02 17

Issues Found – Code Review / Manual Testing 17

High Severity Issues 17

Medium Severity Issues 17

Low Severity Issues 17

Functional Test 18

G. Contract – UnifarmLibrary 19

Issues Found – Code Review / Manual Testing 19

High Severity Issues 19

Medium Severity Issues 19

Low Severity Issues 19

Automated Tests 20

Results: 30

Closing Summary 31

Scope of the Audit

The scope of this audit was to analyze and document the Unifarm smart contracts codebase for quality, security, and correctness.

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
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

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 ERC20 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 analysed 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.

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 the security of smart contracts.

Code Review / Manual Analysis

Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the 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 the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis, Theo.

Issue Categories

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

Risk-level	Description
High	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 be fixed before moving to a live environment.
Medium	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	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 issues that 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
Open	0	0	0	0
Acknowledged	0	0	2	0
Closed	0	0	2	0

Introduction

During the period of **December 7, 2021, to January 11 2022** - QuillAudits Team performed a security audit for **Unifarm** smart contracts.

The code for the audit was taken from the following official repo of **Unifarm**:
https://github.com/themohitmadan/unifarm_amm/tree/development

V	Date	Commit ID/Contract address
1	December	8db8a90c5da254bc2cc03f6fde21c0f00cbcf467
2	January	80c2912fb0119fe41135be75a5de2f8fa107265f

Issues Found – Code Review / Manual Testing

A.Contract - GovernorBravoDelegate

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

A.1 Missing Address Verification

```
Line 166:
        uint256 startBlock = add256(block.number, votingDelay);
        uint256 endBlock = add256(startBlock, votingPeriod);
        proposalCount = add256(proposalCount, 1);
        Proposal memory newProposal = Proposal({
            id: proposalCount,
            token: tokenAddress,
            amount: amount,
            proposer: _msgSender(),
            eta: 0,
            targets: targets,
            values: values,
            signatures: signatures,
            calldatas: calldatas,
            startBlock: startBlock,
            endBlock: endBlock,
            forVotes: 0,
            againstVotes: 0,
            abstainVotes: 0,
            canceled: false,
            executed: false
        });
```

Description

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, the contract's functionality may become inaccessible or tokens may be burned in perpetuity.

Remediation
It's recommended to undertake further validation prior to user-supplied data. The concerns can be resolved by utilizing a whitelist technique or a modifier.

Acknowledged
The Unifarm team has acknowledged the risk knowing the proposal creator might not want the proposal to have a token.

Status: Acknowledged

A.2 Missing Value Verification

```
Line 166:
    uint256 startBlock = add256(block.number, votingDelay);
    uint256 endBlock = add256(startBlock, votingPeriod);
    proposalCount = add256(proposalCount, 1);
    Proposal memory newProposal = Proposal({
        id: proposalCount,
        token: tokenAddress,
        amount: amount,
        proposer: _msgSender(),
        eta: 0,
        targets: targets,
        values: values,
        signatures: signatures,
        calldatas: calldatas,
        startBlock: startBlock,
        endBlock: endBlock,
        forVotes: 0,
        againstVotes: 0,
        abstainVotes: 0,
        canceled: false,
        executed: false
    });
```

Description
Certain functions lack a safety check in the value, the values that are coming from the argument should be verified, otherwise, the contract's functionality may become inaccessible.

Remediation
Add require statement to verify the values that are coming from the arguments.

Acknowledged
The Unifarm team has acknowledged the risk knowing that there are cases where the amount should be null and all of these scenarios are handled.

Status: Acknowledged

A.3 Floating Pragma

```
Line 3:
  pragma solidity >=0.5.16;
```

Description
The contract makes use of the floating-point pragma 0.7.6 Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version that may introduce issues in the contract system.

Remediation
Consider locking the pragma version. It is advised that floating pragma not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status: Fixed
The Team has fixed the issue in
a52ac3ca9f8534fbcc3a2ba908dfe29373fe9f74

Functional Tests

	Test Result	Logical Result	Overall
initialize	Pass	Pass	Pass
updateTokenPermission	Pass	Pass	Pass
propose	Pass	Pass	Pass
queue	Pass	Pass	Pass
execute	Pass	Pass	Pass
cancel	Pass	Pass	Pass
getActions	Pass	Pass	Pass
getReceipt	Pass	Pass	Pass
state	Pass	Pass	Pass
castVote	Pass	Pass	Pass
castVoteBySig	Pass	Pass	Pass
castVoteInternal	Pass	Pass	Pass
_setVotingDelay	Pass	Pass	Pass
_setVotingPeriod	Pass	Pass	Pass
_setProposalThreshold	Pass	Pass	Pass
claimReward	Pass	Pass	Pass
_initiate	Pass	Pass	Pass
_setPendingAdmin	Pass	Pass	Pass
_acceptAdmin	Pass	Pass	Pass

B.Contract - GovernorBravoDelegator

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

No issues were found.

Functional Tests

	Test Result	Logical Result	Overall
_setImplementation	Pass	Pass	Pass
delegateTo	Pass	Pass	Pass

C.Contract - AMMUtility

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

C.1 Missing Value Verification

```
Line 27:
constructor(address payable _feeTo, uint256 _fee, IWETH _weth) public {
        require(_feeTo != address(0), 'ZERO_ADDRESS');
        require(address(_weth) != address(0), 'ZERO_ADDRESS_WETH');

        fee = _fee;
        feeTo = _feeTo;
        WETH = _weth;
        }
```

Description
Certain functions lack a safety check in the value, the values that are coming from the argument should be verified, otherwise, the contract's functionality may become inaccessible.

Remediation
Add require statement to verify the values that are coming from the arguments.

Status: Fixed
The Unifarm team has fixed the issue by setting the fee value statically in d4b10e79db855882384b68cf29e8a47b6ddf8e03.

Functional Tests

	Test Result	Logical Result	Overall
swapTokens()	Pass	Pass	Pass
withdrawToken()	Pass	Pass	Pass
withdrawETH()	Pass	Pass	Pass

D.Contract - UnifarmFactory

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

No issues were found.

Functional Tests

	Test Result	Logical Result	Overall
getPair()	Pass	Pass	Pass
allPairs(uint)	Pass	Pass	Pass
allPairsLength()	Pass	Pass	Pass
feeTo()	Pass	Pass	Pass
feeToSetter()	Pass	Pass	Pass
createPair()	Pass	Pass	Pass
pairConfigs()	Pass	Pass	Pass
setFeeTo()	Pass	Pass	Pass
updateLPFeeConfig()	Pass	Pass	Pass
updateSwapFeeConfig()	Pass	Pass	Pass

E.Contract - UnifarmPair

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

No issues were found.

Functional Tests

MINIMUM_LIQUIDITY()	Pass	Pass	Pass
factory()	Pass	Pass	Pass
token0()	Pass	Pass	Pass
token1()	Pass	Pass	Pass
getReserves()	Pass	Pass	Pass
price0CumulativeLast()	Pass	Pass	Pass
price1CumulativeLast()	Pass	Pass	Pass
kLast()	Pass	Pass	Pass
mint()	Pass	Pass	Pass
burn()	Pass	Pass	Pass
swap()	Pass	Pass	Pass
skim()	Pass	Pass	Pass
sync()	Pass	Pass	Pass
initialize()	Pass	Pass	Pass

F.Contract - UnifarmRouter02

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

No issues were found.

Functional Tests

	Test Result	Logical Result	Overall
addLiquidity	Pass	Pass	Pass
addLiquidityETH	Pass	Pass	Pass
removeLiquidity	Pass	Pass	Pass
removeLiquidityETH	Pass	Pass	Pass
removeLiquidityWithPermit	Pass	Pass	Pass
removeLiquidityETHWithPermit	Pass	Pass	Pass
removeLiquidityETHSupportingFeeOnTransferTokens	Pass	Pass	Pass
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	Pass	Pass	Pass
swapExactTokensForTokens	Pass	Pass	Pass
swapTokensForExactTokens	Pass	Pass	Pass
swapExactETHForTokens	Pass	Pass	Pass
swapTokensForExactETH	Pass	Pass	Pass
swapExactTokensForETH	Pass	Pass	Pass
swapETHForExactTokens	Pass	Pass	Pass
swapExactTokensForTokensSupportingFeeOnTransferTokens	Pass	Pass	Pass
swapExactETHForTokensSupportingFeeOnTransferTokens	Pass	Pass	Pass
swapExactTokensForETHSupportingFeeOnTransferTokens	Pass	Pass	Pass

G.Contract - UnifarmLibrary

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

No issues were found.

Automated Tests

Slither

```
Reentrancy in UnifarmPair.swap(uint256,uint256,address,bytes) (UnifarmPair.sol#200-246):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (UnifarmPair.sol#218)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - _safeTransfer(_token1,to,amount1Out) (UnifarmPair.sol#219)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - IUnifarmCallee(to).unifarmCall(_msgSender(),amount0Out,amount1Out,data) (UnifarmPair.sol#220)
        - toSub0 = _swapFee(token0,amount0In,to) (UnifarmPair.sol#232)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - toSub1 = _swapFee(token1,amount1In,to) (UnifarmPair.sol#233)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        External calls sending eth:
        - toSub0 = _swapFee(token0,amount0In,to) (UnifarmPair.sol#232)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        - toSub1 = _swapFee(token1,amount1In,to) (UnifarmPair.sol#233)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        Event emitted after the call(s):
        - FeeDeducted(swapFee,false,address(0)) (UnifarmPair.sol#265)
                - toSub1 = _swapFee(token1,amount1In,to) (UnifarmPair.sol#233)
        - FeeDeducted(toSubtract,true,_token) (UnifarmPair.sol#271)
                - toSub1 = _swapFee(token1,amount1In,to) (UnifarmPair.sol#233)
        - Swap(_msgSender(),amount0In,amount1In,amount0Out,amount1Out,to) (UnifarmPair.sol#245)
        - Sync(reserve0,reserve1) (UnifarmPair.sol#120)
                - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#244)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

UnifarmERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (UnifarmERC20.sol#124-148) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,Unifarm: EXPIRED) (UnifarmERC20.sol#134)
UnifarmPair._update(uint256,uint256,uint112,uint112) (UnifarmPair.sol#103-121) uses timestamp for comparisons
        Dangerous comparisons:
        - timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0 (UnifarmPair.sol#112)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

BaseRelayRecipient._msgSender() (BaseRelayRecipient.sol#34-45) uses assembly
        - INLINE ASM (BaseRelayRecipient.sol#39-41)
UnifarmERC20.constructor() (UnifarmERC20.sol#27-41) uses assembly
        - INLINE ASM (UnifarmERC20.sol#29-31)
UnifarmFactory.createPair(address,address) (UnifarmFactory.sol#77-104) uses assembly
        - INLINE ASM (UnifarmFactory.sol#84-86)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.5.16', '=0.5.16']
        - =0.5.16 (BaseRelayRecipient.sol#2)
        - =0.5.16 (Ownable.sol#2)
        - =0.5.16 (UnifarmERC20.sol#3)
        - =0.5.16 (UnifarmFactory.sol#3)
```

```
IRelayRecipient._msgSender() (interfaces/IRelayRecipient.sol#23) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Low level call in UnifarmPair._safeTransfer(address,address,uint256) (UnifarmPair.sol#54-61):
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
Low level call in UnifarmPair._safeTransferETH(address,uint256) (UnifarmPair.sol#63-66):
        - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable UnifarmERC20.DOMAIN_SEPARATOR (UnifarmERC20.sol#19) is not in mixedCase
Parameter UnifarmFactory.setFeeTo(address)._feeTo (UnifarmFactory.sol#106) is not in mixedCase
Parameter UnifarmFactory.updateLPFeeConfig(address,address,bool,uint256)._feeInToken (UnifarmFactory.sol#114) is not in mixedCase
Parameter UnifarmFactory.updateLPFeeConfig(address,address,bool,uint256)._fee (UnifarmFactory.sol#115) is not in mixedCase
Parameter UnifarmFactory.updateSwapFeeConfig(address,address,bool,uint256)._feeInToken (UnifarmFactory.sol#133) is not in mixedCase
Parameter UnifarmFactory.updateSwapFeeConfig(address,address,bool,uint256)._fee (UnifarmFactory.sol#134) is not in mixedCase
Parameter UnifarmPair.initialize(address,address,address)._token0 (UnifarmPair.sol#88) is not in mixedCase
Parameter UnifarmPair.initialize(address,address,address)._token1 (UnifarmPair.sol#89) is not in mixedCase
Parameter UnifarmPair.initialize(address,address,address)._trustedForwarder (UnifarmPair.sol#90) is not in mixedCase
Function IUnifarmERC20.DOMAIN_SEPARATOR() (interfaces/IUnifarmERC20.sol#19) is not in mixedCase
Function IUnifarmERC20.PERMIT_TYPEHASH() (interfaces/IUnifarmERC20.sol#20) is not in mixedCase
Function IUnifarmPair.DOMAIN_SEPARATOR() (interfaces/IUnifarmPair.sol#19) is not in mixedCase
Function IUnifarmPair.PERMIT_TYPEHASH() (interfaces/IUnifarmPair.sol#20) is not in mixedCase
Function IUnifarmPair.MINIMUM_LIQUIDITY() (interfaces/IUnifarmPair.sol#38) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable UnifarmPair.swap(uint256,uint256,address,bytes).balance0Adjusted (UnifarmPair.sol#234) is too similar to UnifarmPair.swap(uint256,uint256,address,bytes).balance1Adjusted (UnifarmPair.sol#235)
Variable UnifarmPair.price0CumulativeLast (UnifarmPair.sol#28) is too similar to UnifarmPair.price1CumulativeLast (UnifarmPair.sol#29)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

UnifarmFactory.pairCodeHash() (UnifarmFactory.sol#73-75) uses literals with too many digits:
        - keccak256(bytes)(type(address)(UnifarmPair).creationCode) (UnifarmFactory.sol#74)
UnifarmFactory.createPair(address,address) (UnifarmFactory.sol#77-104) uses literals with too many digits:
        - bytecode = type(address)(UnifarmPair).creationCode (UnifarmFactory.sol#82)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Ownable.sol#50-52)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Ownable.sol#58-61)
increaseAllowance(address,uint256) should be declared external:
        - UnifarmERC20.increaseAllowance(address,uint256) (UnifarmERC20.sol#108-114)
decreaseAllowance(address,uint256) should be declared external:
        - UnifarmERC20.decreaseAllowance(address,uint256) (UnifarmERC20.sol#116-122)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Ownable.sol#50-52)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Ownable.sol#58-61)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

```
Contract locking ether found:
        Contract MultiSigWallet (MultiSigWallet.sol#12-432) has payable functions:
         - MultiSigWallet.fallback() (MultiSigWallet.sol#105-107)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether

MultiSigWallet.external_call(address,uint256,uint256,bytes) (MultiSigWallet.sol#285-308) uses assembly
        - INLINE ASM (MultiSigWallet.sol#292-306)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

SafeMath.mul(uint256,uint256) (libraries/SafeMath.sol#15-17) is never used and should be removed
SafeMath.sub(uint256,uint256) (libraries/SafeMath.sol#11-13) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Parameter MultiSigWallet.changeRequirement(uint256)._required (MultiSigWallet.sol#198) is not in mixedCase
Function MultiSigWallet.external_call(address,uint256,uint256,bytes) (MultiSigWallet.sol#285-308) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

addOwner(address) should be declared external:
        - MultiSigWallet.addOwner(address) (MultiSigWallet.sol#136-146)
removeOwner(address) should be declared external:
        - MultiSigWallet.removeOwner(address) (MultiSigWallet.sol#152-166)
replaceOwner(address,address) should be declared external:
        - MultiSigWallet.replaceOwner(address,address) (MultiSigWallet.sol#175-190)
submitTransaction(address,uint256,bytes) should be declared external:
        - MultiSigWallet.submitTransaction(address,uint256,bytes) (MultiSigWallet.sol#212-219)
revokeConfirmation(uint256) should be declared external:
        - MultiSigWallet.revokeConfirmation(uint256) (MultiSigWallet.sol#240-248)
getConfirmationCount(uint256) should be declared external:
        - MultiSigWallet.getConfirmationCount(uint256) (MultiSigWallet.sol#364-366)
getTransactionCount(bool,bool) should be declared external:
        - MultiSigWallet.getTransactionCount(bool,bool) (MultiSigWallet.sol#374-377)
getOwners() should be declared external:
        - MultiSigWallet.getOwners() (MultiSigWallet.sol#383-385)
getConfirmations(uint256) should be declared external:
        - MultiSigWallet.getConfirmations(uint256) (MultiSigWallet.sol#392-403)
getTransactionIds(uint256,uint256,bool,bool) should be declared external:
        - MultiSigWallet.getTransactionIds(uint256,uint256,bool,bool) (MultiSigWallet.sol#415-431)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

UnifarmPair._safeTransferETH(address,uint256) (UnifarmPair.sol#63-66) sends eth to arbitrary user
        Dangerous calls:
        - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

UnifarmPair._update(uint256,uint256,uint112,uint112) (UnifarmPair.sol#103-121) uses a weak PRNG: "blockTimestamp = uint32(block.timestamp % 2 ** 32) (UnifarmPair.sol#110)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG
```

```
Reentrancy in UnifarmPair.burn(address) (UnifarmPair.sol#170-197):
        External calls:
        - _safeTransfer(_token0,to,amount0) (UnifarmPair.sol#187)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - _safeTransfer(_token1,to,amount1) (UnifarmPair.sol#188)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - _safeTransferETH(to,ethFee) (UnifarmPair.sol#189)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        External calls sending eth:
        - _safeTransferETH(to,ethFee) (UnifarmPair.sol#189)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#194)
                - blockTimestampLast = blockTimestamp (UnifarmPair.sol#119)
        - kLast = uint256(reserve0).mul(reserve1) (UnifarmPair.sol#195)
        - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#194)
                - reserve0 = uint112(balance0) (UnifarmPair.sol#117)
        - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#194)
                - reserve1 = uint112(balance1) (UnifarmPair.sol#118)
Reentrancy in UnifarmPair.swap(uint256,uint256,address,bytes) (UnifarmPair.sol#200-246):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (UnifarmPair.sol#218)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - _safeTransfer(_token1,to,amount1Out) (UnifarmPair.sol#219)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - IUnifarmCallee(to).unifarmCall(_msgSender(),amount0Out,amount1Out,data) (UnifarmPair.sol#220)
        - toSub0 = _swapFee(token0,amount0In,to) (UnifarmPair.sol#232)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - toSub1 = _swapFee(token1,amount1In,to) (UnifarmPair.sol#233)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        External calls sending eth:
        - toSub0 = _swapFee(token0,amount0In,to) (UnifarmPair.sol#232)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        - toSub1 = _swapFee(token1,amount1In,to) (UnifarmPair.sol#233)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#244)
                - blockTimestampLast = blockTimestamp (UnifarmPair.sol#119)
        - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#244)
                - reserve0 = uint112(balance0) (UnifarmPair.sol#117)
        - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#244)
                - reserve1 = uint112(balance1) (UnifarmPair.sol#118)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

UnifarmPair._swapFee(address,uint256,address) (UnifarmPair.sol#248-278) performs a multiplication on the result of a division:
        -toSubtract = _amount.mul(swapFee) / (1000) (UnifarmPair.sol#269)
        -toSubtract = toSubtract.mul(lpFee) (UnifarmPair.sol#276)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
```

```
UnifarmPair._safeTransfer(address,address,uint256) (UnifarmPair.sol#54-61) uses a dangerous strict equality:
        - require(bool,string)(success && (data.length == 0 || abi.decode(data,(bool))),Unifarm: TRANSFER_FAILED) (UnifarmPair.sol#60)
UnifarmPair.mint(address) (UnifarmPair.sol#145-167) uses a dangerous strict equality:
        - _totalSupply == 0 (UnifarmPair.sol#155)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

UnifarmPair.swap(uint256,uint256,address,bytes).toSub0 (UnifarmPair.sol#230) is a local variable never initialized
UnifarmPair.swap(uint256,uint256,address,bytes).toSub1 (UnifarmPair.sol#231) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

UnifarmPair.initialize(address,address,address) (UnifarmPair.sol#87-100) should emit an event for:
        - trustedForwarder = _trustedForwarder (UnifarmPair.sol#99)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

Reentrancy in UnifarmPair.burn(address) (UnifarmPair.sol#170-197):
        External calls:
        - _safeTransfer(_token0,to,amount0) (UnifarmPair.sol#187)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - _safeTransfer(_token1,to,amount1) (UnifarmPair.sol#188)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - _safeTransferETH(to,ethFee) (UnifarmPair.sol#189)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        External calls sending eth:
        - _safeTransferETH(to,ethFee) (UnifarmPair.sol#189)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#194)
                - price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (UnifarmPair.sol#114)
        - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#194)
                - price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (UnifarmPair.sol#115)
Reentrancy in UnifarmPair.swap(uint256,uint256,address,bytes) (UnifarmPair.sol#200-246):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (UnifarmPair.sol#218)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - _safeTransfer(_token1,to,amount1Out) (UnifarmPair.sol#219)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - IUnifarmCallee(to).unifarmCall(_msgSender(),amount0Out,amount1Out,data) (UnifarmPair.sol#220)
        - toSub0 = _swapFee(token0,amount0In,to) (UnifarmPair.sol#232)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - toSub1 = _swapFee(token1,amount1In,to) (UnifarmPair.sol#233)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        External calls sending eth:
        - toSub0 = _swapFee(token0,amount0In,to) (UnifarmPair.sol#232)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        - toSub1 = _swapFee(token1,amount1In,to) (UnifarmPair.sol#233)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#244)
```

```
Reentrancy in UnifarmPair._swapFee(address,uint256,address) (UnifarmPair.sol#248-278):
        External calls:
        - _safeTransferETH(to,swapFee) (UnifarmPair.sol#264)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        Event emitted after the call(s):
        - FeeDeducted(swapFee,false,address(0)) (UnifarmPair.sol#265)
Reentrancy in UnifarmPair._swapFee(address,uint256,address) (UnifarmPair.sol#248-278):
        External calls:
        - _safeTransferETH(to,swapFee) (UnifarmPair.sol#264)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        - _safeTransfer(_token,feeTo,toSubtract) (UnifarmPair.sol#270)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        External calls sending eth:
        - _safeTransferETH(to,swapFee) (UnifarmPair.sol#264)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        Event emitted after the call(s):
        - FeeDeducted(toSubtract,true,_token) (UnifarmPair.sol#271)
Reentrancy in UnifarmPair.burn(address) (UnifarmPair.sol#170-197):
        External calls:
        - _safeTransfer(_token0,to,amount0) (UnifarmPair.sol#187)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - _safeTransfer(_token1,to,amount1) (UnifarmPair.sol#188)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - _safeTransferETH(to,ethFee) (UnifarmPair.sol#189)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        External calls sending eth:
        - _safeTransferETH(to,ethFee) (UnifarmPair.sol#189)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        Event emitted after the call(s):
        - Burn(_msgSender(),amount0,amount1,to,ethFee) (UnifarmPair.sol#196)
        - Sync(reserve0,reserve1) (UnifarmPair.sol#120)
                - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#194)
Reentrancy in UnifarmPair.swap(uint256,uint256,address,bytes) (UnifarmPair.sol#200-246):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (UnifarmPair.sol#218)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - _safeTransfer(_token1,to,amount1Out) (UnifarmPair.sol#219)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - IUnifarmCallee(to).unifarmCall(_msgSender(),amount0Out,amount1Out,data) (UnifarmPair.sol#220)
        - toSub0 = _swapFee(token0,amount0In,to) (UnifarmPair.sol#232)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        External calls sending eth:
        - toSub0 = _swapFee(token0,amount0In,to) (UnifarmPair.sol#232)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        Event emitted after the call(s):
        - FeeDeducted(swapFee,false,address(0)) (UnifarmPair.sol#265)
                - toSub0 = _swapFee(token0,amount0In,to) (UnifarmPair.sol#232)
        - FeeDeducted(toSubtract,true,_token) (UnifarmPair.sol#271)
                - toSub0 = _swapFee(token0,amount0In,to) (UnifarmPair.sol#232)
```

```
Reentrancy in UnifarmPair.swap(uint256,uint256,address,bytes) (UnifarmPair.sol#200-246):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (UnifarmPair.sol#218)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - _safeTransfer(_token1,to,amount1Out) (UnifarmPair.sol#219)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - IUnifarmCallee(to).unifarmCall(_msgSender(),amount0Out,amount1Out,data) (UnifarmPair.sol#220)
        - toSub0 = _swapFee(token0,amount0In,to) (UnifarmPair.sol#232)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - toSub1 = _swapFee(token1,amount1In,to) (UnifarmPair.sol#233)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        External calls sending eth:
        - toSub0 = _swapFee(token0,amount0In,to) (UnifarmPair.sol#232)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        - toSub1 = _swapFee(token1,amount1In,to) (UnifarmPair.sol#233)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        Event emitted after the call(s):
        - FeeDeducted(swapFee,false,address(0)) (UnifarmPair.sol#265)
                - toSub1 = _swapFee(token1,amount1In,to) (UnifarmPair.sol#233)
        - FeeDeducted(toSubtract,true,_token) (UnifarmPair.sol#271)
                - toSub1 = _swapFee(token1,amount1In,to) (UnifarmPair.sol#233)
        - Swap(_msgSender(),amount0In,amount1In,amount0Out,amount1Out,to) (UnifarmPair.sol#245)
        - Sync(reserve0,reserve1) (UnifarmPair.sol#120)
                - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#244)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

UnifarmERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (UnifarmERC20.sol#124-148) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,Unifarm: EXPIRED) (UnifarmERC20.sol#134)
UnifarmPair._update(uint256,uint256,uint112,uint112) (UnifarmPair.sol#103-121) uses timestamp for comparisons
        Dangerous comparisons:
        - timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0 (UnifarmPair.sol#112)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

BaseRelayRecipient._msgSender() (BaseRelayRecipient.sol#34-45) uses assembly
        - INLINE ASM (BaseRelayRecipient.sol#39-41)
UnifarmERC20.constructor() (UnifarmERC20.sol#27-41) uses assembly
        - INLINE ASM (UnifarmERC20.sol#29-31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Different versions of Solidity is used:
        - Version used: ['0.5.16', '=0.5.16']
        - =0.5.16 (BaseRelayRecipient.sol#2)
        - =0.5.16 (UnifarmERC20.sol#3)
        - =0.5.16 (UnifarmPair.sol#3)
        - 0.5.16 (interfaces/IERC20.sol#2)
        - =0.5.16 (interfaces/IRelayRecipient.sol#2)
        - =0.5.16 (interfaces/IUnifarmCallee.sol#2)
        - =0.5.16 (interfaces/IUnifarmERC20.sol#2)
        - =0.5.16 (interfaces/IUnifarmFactory.sol#2)
        - =0.5.16 (interfaces/IUnifarmPair.sol#2)
        - =0.5.16 (libraries/Math.sol#3)
        - =0.5.16 (libraries/SafeMath.sol#2)
        - =0.5.16 (libraries/UQ112x112.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

IRelayRecipient._msgSender() (interfaces/IRelayRecipient.sol#23) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Low level call in UnifarmPair._safeTransfer(address,address,uint256) (UnifarmPair.sol#54-61):
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
Low level call in UnifarmPair._safeTransferETH(address,uint256) (UnifarmPair.sol#63-66):
        - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable UnifarmERC20.DOMAIN_SEPARATOR (UnifarmERC20.sol#19) is not in mixedCase
Parameter UnifarmPair.initialize(address,address,address)._token0 (UnifarmPair.sol#88) is not in mixedCase
Parameter UnifarmPair.initialize(address,address,address)._token1 (UnifarmPair.sol#89) is not in mixedCase
Parameter UnifarmPair.initialize(address,address,address)._trustedForwarder (UnifarmPair.sol#90) is not in mixedCase
Function IUnifarmERC20.DOMAIN_SEPARATOR() (interfaces/IUnifarmERC20.sol#19) is not in mixedCase
Function IUnifarmERC20.PERMIT_TYPEHASH() (interfaces/IUnifarmERC20.sol#20) is not in mixedCase
Function IUnifarmPair.DOMAIN_SEPARATOR() (interfaces/IUnifarmPair.sol#19) is not in mixedCase
Function IUnifarmPair.PERMIT_TYPEHASH() (interfaces/IUnifarmPair.sol#20) is not in mixedCase
Function IUnifarmPair.MINIMUM_LIQUIDITY() (interfaces/IUnifarmPair.sol#38) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable UnifarmPair.swap(uint256,uint256,address,bytes).balance0Adjusted (UnifarmPair.sol#234) is too similar to UnifarmPair.swap(uint256,uint256,address,bytes).balance1Adjusted (UnifarmPair.sol#235)
Variable UnifarmPair.price0CumulativeLast (UnifarmPair.sol#28) is too similar to UnifarmPair.price1CumulativeLast (UnifarmPair.sol#29)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

increaseAllowance(address,uint256) should be declared external:
        - UnifarmERC20.increaseAllowance(address,uint256) (UnifarmERC20.sol#108-114)
decreaseAllowance(address,uint256) should be declared external:
        - UnifarmERC20.decreaseAllowance(address,uint256) (UnifarmERC20.sol#116-122)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

BaseRelayRecipient.trustedForwarder (BaseRelayRecipient.sol#14) is never initialized. It is used in:
        - BaseRelayRecipient.isTrustedForwarder(address) (BaseRelayRecipient.sol#24-26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
```

```
BaseRelayRecipient._msgSender() (BaseRelayRecipient.sol#34-45) uses assembly
        - INLINE ASM (BaseRelayRecipient.sol#39-41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

BaseRelayRecipient._msgSender() (BaseRelayRecipient.sol#34-45) is never used and should be removed
IRelayRecipient._msgSender() (interfaces/IRelayRecipient.sol#23) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

BaseRelayRecipient (BaseRelayRecipient.sol#10-46) does not implement functions:
        - IRelayRecipient.versionRecipient() (interfaces/IRelayRecipient.sol#25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

BaseRelayRecipient.trustedForwarder (BaseRelayRecipient.sol#14) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

BaseRelayRecipient.trustedForwarder (BaseRelayRecipient.sol#14) is never initialized. It is used in:
        - BaseRelayRecipient.isTrustedForwarder(address) (BaseRelayRecipient.sol#24-26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

UnifarmERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (UnifarmERC20.sol#124-148) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,Unifarm: EXPIRED) (UnifarmERC20.sol#134)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

BaseRelayRecipient._msgSender() (BaseRelayRecipient.sol#34-45) uses assembly
        - INLINE ASM (BaseRelayRecipient.sol#39-41)
UnifarmERC20.constructor() (UnifarmERC20.sol#27-41) uses assembly
        - INLINE ASM (UnifarmERC20.sol#29-31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

IRelayRecipient._msgSender() (interfaces/IRelayRecipient.sol#23) is never used and should be removed
SafeMath.mul(uint256,uint256) (libraries/SafeMath.sol#15-17) is never used and should be removed
UnifarmERC20._burn(address,uint256) (UnifarmERC20.sol#58-62) is never used and should be removed
UnifarmERC20._mint(address,uint256) (UnifarmERC20.sol#52-56) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Variable UnifarmERC20.DOMAIN_SEPARATOR (UnifarmERC20.sol#19) is not in mixedCase
Function IUnifarmERC20.DOMAIN_SEPARATOR() (interfaces/IUnifarmERC20.sol#19) is not in mixedCase
Function IUnifarmERC20.PERMIT_TYPEHASH() (interfaces/IUnifarmERC20.sol#20) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

BaseRelayRecipient.trustedForwarder (BaseRelayRecipient.sol#14) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

increaseAllowance(address,uint256) should be declared external:
        - UnifarmERC20.increaseAllowance(address,uint256) (UnifarmERC20.sol#108-114)
decreaseAllowance(address,uint256) should be declared external:
        - UnifarmERC20.decreaseAllowance(address,uint256) (UnifarmERC20.sol#116-122)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

```
UnifarmPair._safeTransferETH(address,uint256) (UnifarmPair.sol#63-66) sends eth to arbitrary user
        Dangerous calls:
        - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

UnifarmPair._update(uint256,uint256,uint112,uint112) (UnifarmPair.sol#103-121) uses a weak PRNG: "blockTimestamp = uint32(block.timestamp % 2 ** 32) (UnifarmPair.sol#110)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

Reentrancy in UnifarmPair.burn(address) (UnifarmPair.sol#170-197):
        External calls:
        - _safeTransfer(_token0,to,amount0) (UnifarmPair.sol#187)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - _safeTransfer(_token1,to,amount1) (UnifarmPair.sol#188)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - _safeTransferETH(to,ethFee) (UnifarmPair.sol#189)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        External calls sending eth:
        - _safeTransferETH(to,ethFee) (UnifarmPair.sol#189)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#194)
                - blockTimestampLast = blockTimestamp (UnifarmPair.sol#119)
        - kLast = uint256(reserve0).mul(reserve1) (UnifarmPair.sol#195)
        - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#194)
                - reserve0 = uint112(balance0) (UnifarmPair.sol#117)
        - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#194)
                - reserve1 = uint112(balance1) (UnifarmPair.sol#118)
Reentrancy in UnifarmPair.swap(uint256,uint256,address,bytes) (UnifarmPair.sol#200-246):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (UnifarmPair.sol#218)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - _safeTransfer(_token1,to,amount1Out) (UnifarmPair.sol#219)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - IUnifarmCallee(to).unifarmCall(_msgSender(),amount0Out,amount1Out,data) (UnifarmPair.sol#220)
        - toSub0 = _swapFee(token0,amount0In,to) (UnifarmPair.sol#232)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - toSub1 = _swapFee(token1,amount1In,to) (UnifarmPair.sol#233)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        External calls sending eth:
        - toSub0 = _swapFee(token0,amount0In,to) (UnifarmPair.sol#232)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        - toSub1 = _swapFee(token1,amount1In,to) (UnifarmPair.sol#233)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#244)
                - blockTimestampLast = blockTimestamp (UnifarmPair.sol#119)
        - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#244)
                - reserve0 = uint112(balance0) (UnifarmPair.sol#117)
```

```
UnifarmPair._swapFee(address,uint256,address) (UnifarmPair.sol#248-278) performs a multiplication on the result of a division:
        -toSubtract = _amount.mul(swapFee) / (1000) (UnifarmPair.sol#269)
        -toSubtract = toSubtract.mul(lpFee) (UnifarmPair.sol#276)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

UnifarmPair._safeTransfer(address,address,uint256) (UnifarmPair.sol#54-61) uses a dangerous strict equality:
        - require(bool,string)(success && (data.length == 0 || abi.decode(data,(bool))),Unifarm: TRANSFER_FAILED) (UnifarmPair.sol#60)
UnifarmPair.mint(address) (UnifarmPair.sol#145-167) uses a dangerous strict equality:
        - _totalSupply == 0 (UnifarmPair.sol#155)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in UnifarmFactory.createPair(address,address) (UnifarmFactory.sol#77-104):
        External calls:
        - IUnifarmPair(pair).initialize(token0,token1,trustedForwarder) (UnifarmFactory.sol#87)
        State variables written after the call(s):
        - getPair[token0][token1] = pair (UnifarmFactory.sol#88)
        - getPair[token1][token0] = pair (UnifarmFactory.sol#89)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

UnifarmFactory.updateLPFeeConfig(address,address,bool,uint256) (UnifarmFactory.sol#111-128) contains a tautology or contradiction:
        - require(bool,string)(_fee >= 0 && _fee <= 1000,Unifarm: INVALID_FEE) (UnifarmFactory.sol#120)
UnifarmFactory.updateSwapFeeConfig(address,address,bool,uint256) (UnifarmFactory.sol#130-147) contains a tautology or contradiction:
        - require(bool,string)(_fee >= 0 && _fee <= 1000,Unifarm: INVALID_FEE) (UnifarmFactory.sol#139)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-contradiction

UnifarmPair.swap(uint256,uint256,address,bytes).toSub1 (UnifarmPair.sol#231) is a local variable never initialized
UnifarmPair.swap(uint256,uint256,address,bytes).toSub0 (UnifarmPair.sol#230) is a local variable never initialized
UnifarmFactory.constructor(address,address,uint256,uint256,bool,bool).pair (UnifarmFactory.sol#49) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

UnifarmPair.initialize(address,address,address) (UnifarmPair.sol#87-100) should emit an event for:
        - trustedForwarder = _trustedForwarder (UnifarmPair.sol#99)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

Reentrancy in UnifarmPair.burn(address) (UnifarmPair.sol#170-197):
        External calls:
        - _safeTransfer(_token0,to,amount0) (UnifarmPair.sol#187)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - _safeTransfer(_token1,to,amount1) (UnifarmPair.sol#188)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - _safeTransferETH(to,ethFee) (UnifarmPair.sol#189)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        External calls sending eth:
        - _safeTransferETH(to,ethFee) (UnifarmPair.sol#189)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#194)
                - price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (UnifarmPair.sol#114)
        - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#194)
                - price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (UnifarmPair.sol#115)
```

```
Reentrancy in UnifarmFactory.createPair(address,address) (UnifarmFactory.sol#77-104):
        External calls:
        - IUnifarmPair(pair).initialize(token0,token1,trustedForwarder) (UnifarmFactory.sol#87)
        State variables written after the call(s):
        - allPairs.push(pair) (UnifarmFactory.sol#94)
        - pairConfigs[pair] = globalConfig (UnifarmFactory.sol#92)
Reentrancy in UnifarmPair.swap(uint256,uint256,address,bytes) (UnifarmPair.sol#200-246):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (UnifarmPair.sol#218)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - _safeTransfer(_token1,to,amount1Out) (UnifarmPair.sol#219)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - IUnifarmCallee(to).unifarmCall(_msgSender(),amount0Out,amount1Out,data) (UnifarmPair.sol#220)
        - toSub0 = _swapFee(token0,amount0In,to) (UnifarmPair.sol#232)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - toSub1 = _swapFee(token1,amount1In,to) (UnifarmPair.sol#233)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        External calls sending eth:
        - toSub0 = _swapFee(token0,amount0In,to) (UnifarmPair.sol#232)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        - toSub1 = _swapFee(token1,amount1In,to) (UnifarmPair.sol#233)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#244)
                - price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (UnifarmPair.sol#114)
        - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#244)
                - price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (UnifarmPair.sol#115)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in UnifarmPair._swapFee(address,uint256,address) (UnifarmPair.sol#248-278):
        External calls:
        - _safeTransferETH(to,swapFee) (UnifarmPair.sol#264)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        Event emitted after the call(s):
        - FeeDeducted(swapFee,false,address(0)) (UnifarmPair.sol#265)
Reentrancy in UnifarmPair._swapFee(address,uint256,address) (UnifarmPair.sol#248-278):
        External calls:
        - _safeTransferETH(to,swapFee) (UnifarmPair.sol#264)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        - _safeTransfer(_token,feeTo,toSubtract) (UnifarmPair.sol#270)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        External calls sending eth:
        - _safeTransferETH(to,swapFee) (UnifarmPair.sol#264)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        Event emitted after the call(s):
        - FeeDeducted(toSubtract,true,_token) (UnifarmPair.sol#271)
```

```
Reentrancy in UnifarmPair.burn(address) (UnifarmPair.sol#170-197):
        External calls:
        - _safeTransfer(_token0,to,amount0) (UnifarmPair.sol#187)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - _safeTransfer(_token1,to,amount1) (UnifarmPair.sol#188)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - _safeTransferETH(to,ethFee) (UnifarmPair.sol#189)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        External calls sending eth:
        - _safeTransferETH(to,ethFee) (UnifarmPair.sol#189)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        Event emitted after the call(s):
        - Burn(_msgSender(),amount0,amount1,to,ethFee) (UnifarmPair.sol#196)
        - Sync(reserve0,reserve1) (UnifarmPair.sol#120)
                - _update(balance0,balance1,_reserve0,_reserve1) (UnifarmPair.sol#194)
Reentrancy in UnifarmFactory.createPair(address,address) (UnifarmFactory.sol#77-104):
        External calls:
        - IUnifarmPair(pair).initialize(token0,token1,trustedForwarder) (UnifarmFactory.sol#87)
        Event emitted after the call(s):
        - PairCreated(token0,token1,pair,globalConfig.lpFeesInToken,globalConfig.swapFeesInToken,globalConfig.lpFee,globalConfig.swapFee) (UnifarmFactory.sol#95-103)
Reentrancy in UnifarmPair.swap(uint256,uint256,address,bytes) (UnifarmPair.sol#200-246):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (UnifarmPair.sol#218)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - _safeTransfer(_token1,to,amount1Out) (UnifarmPair.sol#219)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - IUnifarmCallee(to).unifarmCall(_msgSender(),amount0Out,amount1Out,data) (UnifarmPair.sol#220)
        - toSub0 = _swapFee(token0,amount0In,to) (UnifarmPair.sol#232)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        External calls sending eth:
        - toSub0 = _swapFee(token0,amount0In,to) (UnifarmPair.sol#232)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
        Event emitted after the call(s):
        - FeeDeducted(swapFee,false,address(0)) (UnifarmPair.sol#265)
                - toSub0 = _swapFee(token0,amount0In,to) (UnifarmPair.sol#232)
        - FeeDeducted(toSubtract,true,_token) (UnifarmPair.sol#271)
                - toSub0 = _swapFee(token0,amount0In,to) (UnifarmPair.sol#232)
Reentrancy in UnifarmPair.swap(uint256,uint256,address,bytes) (UnifarmPair.sol#200-246):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (UnifarmPair.sol#218)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - _safeTransfer(_token1,to,amount1Out) (UnifarmPair.sol#219)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - IUnifarmCallee(to).unifarmCall(_msgSender(),amount0Out,amount1Out,data) (UnifarmPair.sol#220)
        - toSub0 = _swapFee(token0,amount0In,to) (UnifarmPair.sol#232)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (UnifarmPair.sol#59)
        - toSub1 = _swapFee(token1,amount1In,to) (UnifarmPair.sol#233)
                - (success) = to.call.value(value)(new bytes(0)) (UnifarmPair.sol#64)
```

Results

No major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorized above according to their level of severity.

Closing Summary

Overall, smart contracts are very well written and adhere to guidelines. Many issues were discovered during the audit; the majority of them are fixed.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the **Unifarm** Contracts. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the **Unifarm** Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Audit Report
January, 2022

For

UniFarm
by OroPocket

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com