Quantstamp Security Assessment Certificate

March 23rd 2022 — Quantstamp Verified

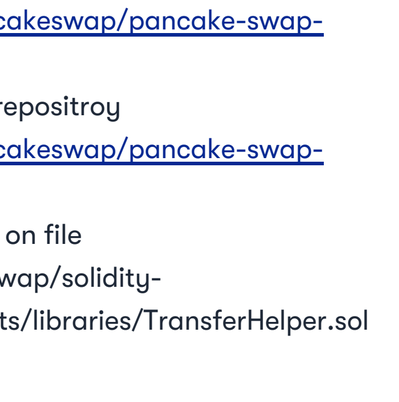
ArthSwap

This audit report was prepared by Quantstamp, the leader in blockchain security.

Executive Summary

Type	Automated Market Maker
Auditors	Roman Rohleder, Research Engineer Alex Murashkin, Senior Software Engineer Sung-Shine Lee, Research Engineer
Timeline	2022-01-25 through 2022-02-01
EVM	Arrow Glacier
Languages	Solidity
Methods	Architecture Review, Manual Review
Specification	None
Documentation Quality	Medium
Test Quality	Medium
Diff/Fork information	This code was based on the repository https://github.com/pancakeswap/pancake-swap-core/ at commit 3b21430, on repositroy https://github.com/pancakeswap/pancake-swap-periphery at commit d769a6d and on file https://github.com/Uniswap/solidity-lib/blob/master/contracts/libraries/TransferHelper.sol at commit c01640b.
Source Code	Repository: marged-sols — Commit: 3db3d7f Repository: arthswap-core — Commit: 7e1b47e
Goals	• Make sure changes made to not pose a security risk.

Total Issues	**6** (5 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	**1** (1 Resolved)
Low Risk Issues	**2** (2 Resolved)
Informational Risk Issues	**3** (2 Resolved)
Undetermined Risk Issues	0 (0 Resolved)



ALL ISSUES ADDRESSED

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
Undetermined	The impact of the issue is uncertain.
Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
Acknowledged	The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).
Resolved	Adjusted program implementation, requirements or constraints to eliminate the risk.
Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

The audited project consisted of a fork of PancakeSwap, with fee values as coming from UniswapV2 and a custom contract `MultiCall.sol`. During the audit we have found issues spanning from medium to informational severity levels. We recommend addressing all the issues. The provided repository only contained merged contracts, missing any tests or deployment scripts, we could therefore neither run any tests nor compute tests coverages. We highly recommend to (re-)add tests and deployment scripts.
After Re-Audit: All of the 6 findings have been addressed by the developers, by fixing 5 of 6 and acknowledging the remaining one. Further, the team added tests (as based on Uniswap and PancakeSwap tests), all of which were passing. The suggested best practices have not been addressed.

ID	Description	Severity	Status
QSP-1	Repository does not contain any test or deployment scripts	^ Medium	Fixed
QSP-2	Use of deprecated `PancakeRouter01.sol`	⌄ Low	Fixed
QSP-3	Privileged Roles and Ownership	⌄ Low	Fixed
QSP-4	Allowance Double-Spend Exploit	○ Informational	Acknowledged
QSP-5	Unlocked Pragma	○ Informational	Fixed
QSP-6	Limitation of `blockhash()`	○ Informational	Fixed

Quantstamp Audit Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:

- Slither v0.8.2

Steps taken to run the tools:

1. Installed the Slither tool: `pip install slither-analyzer`
2. Run Slither from the project directory: `slither .`

Findings

QSP-1 Repository does not contain any test or deployment scripts

Severity: *Medium Risk*

Status: Fixed

Description: Even if the DeFi contracts are themselves secure, the actual security often depends on whether the deployer knows how to configure them properly. Hence a test suite with basic tests is important to cover the basic cases and verify especially the changes.

Recommendation: Write up tests and confirm, for example, the fees are being taken properly as expected.

Update: The repository has been restructured and tests have been added. The missing deployment scripts have been acknowledged, since the contracts have been already deployed.

QSP-2 Use of deprecated `PancakeRouter01.sol`

Severity: *Low Risk*

Status: Fixed

File(s) affected: `PancakeRouter01.sol`

Description: Contract `PancakeRouter.sol` is an improved version of `PancakeRouter01.sol`, adding functions for handling fee-on-transfer tokens and fixing a bug in `PancakeRouter01.getAmountIn()`, which wrongfully was calling `PancakeLibrary.getAmountOut()`.

Recommendation: Remove `PancakeRouter01.sol` and only keep and use the newer `PancakeRouter.sol`.

Update: Fixed, by dropping `PancakeRouter01.sol`, as suggested.

QSP-3 Privileged Roles and Ownership

Severity: *Low Risk*

Status: Fixed

File(s) affected: `PancakeFactory.sol`

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract. The contract `PancakeFactory.sol` contains the following privileged roles:

- `feeToSetter`, as set during the constructor or later by `PancakeFactory.setFeeToSetter()`:
  . Assign or reset the `feeTo` recipient, which would activate/deactivate sending 16.6% of the 0.3% trading fees (0.05%) to that address, by calling `PancakeFactory.setFeeTo()`.
  . Assign a new `feeToSetter` role or renounce it forever, by calling `PancakeFactory.setFeeToSetter()`.

Recommendation: This centralization of power needs to be made clear to the users.

Update: Fixed, by adding corresponding information on public facing documentation.

QSP-4 Allowance Double-Spend Exploit

Severity: *Informational*

Status: Acknowledged

File(s) affected: `PancakeFactory.sol (PancakeERC20)`

Description: As it presently is constructed, the contract is vulnerable to the allowance double-spend exploit, as with other ERC20 tokens.

Exploit Scenario:

1. Alice allows Bob to transfer `N` amount of Alice's tokens (`N>0`) by calling the `approve()` method on `Token` smart contract (passing Bob's address and `N` as method arguments)
2. After some time, Alice decides to change from `N` to `M` (`M>0`) the number of Alice's tokens Bob is allowed to transfer, so she calls the `approve()` method again, this time passing Bob's address and `M` as method arguments
3. Bob notices Alice's second transaction before it was mined and quickly sends another transaction that calls the `transferFrom()` method to transfer `N` Alice's tokens somewhere
4. If Bob's transaction will be executed before Alice's transaction, then Bob will successfully transfer `N` Alice's tokens and will gain an ability to transfer another `M` tokens
5. Before Alice notices any irregularities, Bob calls `transferFrom()` method again, this time to transfer `M` Alice's tokens.

Recommendation: The exploit (as described above) is mitigated through use of functions that increase/decrease the allowance relative to its current value, such as `increaseAllowance()` and `decreaseAllowance()`.
Pending community agreement on an ERC standard that would protect against this exploit, we recommend that developers of applications dependent on `approve()` / `transferFrom()` should keep in mind that they have to set allowance to 0 first and verify if it was used before setting the new value. Teams who decide to wait for such a standard should make these recommendations to app developers who work with their token contract.

Update: This finding and its implications have been acknowledged by the developers.

QSP-5 Unlocked Pragma

Severity: *Informational*

Status: Fixed

File(s) affected: `MultiCall.sol`

Related Issue(s): SWC-103

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.4.*`. The caret (^) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version *and above*, hence the term "unlocked".

Recommendation: For consistency and to prevent unexpected behavior in the future, it is recommended to remove the caret to lock the file onto a specific Solidity version.

Update: Fixed, by removing `MultiCall.sol` altogether.

QSP-6 Limitation of `blockhash()`

Severity: *Informational*

Status: Fixed

File(s) affected: `MultiCall.sol`

Description: The solidity built-in function `getblockhash()` only works for the last 256 most recent blocks, excluding the current one and otherwise returns zero. Function `MultiCall.getBlockHash()` is correspondingly limited.

Recommendation: Keep this limitation in mind when designing the front-end and/or communicate this limitation in user-facing public documentation.

Update: Fixed, by removing `MultiCall.sol` altogether.

Automated Analyses

Slither

We have run the latest version of the slither analyzer on this repository's solidity code. The tool has identified 117 issues in total, most of which were filtered out as false positives. The remaining issues were integrated in the findings of this report.

```
slither MultiCall.sol

Multicall.aggregate(Multicall.Call[]) (MultiCall.sol#20-28) has external calls inside a loop: (success,ret) = calls[i].target.call(calls[i].callData) (MultiCall.sol#24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Pragma version>=0.5.0 (MultiCall.sol#6) 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

Low level call in Multicall.aggregate(Multicall.Call[]) (MultiCall.sol#20-28):
    - (success,ret) = calls[i].target.call(calls[i].callData) (MultiCall.sol#24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

aggregate(Multicall.Call[]) should be declared external:
    - Multicall.aggregate(Multicall.Call[]) (MultiCall.sol#20-28)
getEthBalance(address) should be declared external:
    - Multicall.getEthBalance(address) (MultiCall.sol#31-33)
getBlockHash(uint256) should be declared external:
    - Multicall.getBlockHash(uint256) (MultiCall.sol#35-37)
getLastBlockHash() should be declared external:
    - Multicall.getLastBlockHash() (MultiCall.sol#39-41)
getCurrentBlockTimestamp() should be declared external:
    - Multicall.getCurrentBlockTimestamp() (MultiCall.sol#43-45)
getCurrentBlockDifficulty() should be declared external:
    - Multicall.getCurrentBlockDifficulty() (MultiCall.sol#47-49)
getCurrentBlockGasLimit() should be declared external:
    - Multicall.getCurrentBlockGasLimit() (MultiCall.sol#51-53)
getCurrentBlockCoinbase() should be declared external:
    - Multicall.getCurrentBlockCoinbase() (MultiCall.sol#55-57)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
MultiCall.sol analyzed (1 contracts with 77 detectors), 12 result(s) found

slither PancakeRouter01.sol

PancakeRouter01.removeLiquidity(address,address,uint256,uint256,uint256,address,uint256) (PancakeRouter01.sol#394-410) ignores return value by IPancakePair(pair).transferFrom(msg.sender,pair,liquidity)
(PancakeRouter01.sol#404)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

PancakeLibrary.getAmountsOut(address,uint256,address[]).i (PancakeRouter01.sol#172) is a local variable never initialized
PancakeRouter01._swap(uint256[],address[],address).i (PancakeRouter01.sol#465) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

PancakeRouter01._addLiquidity(address,address,uint256,uint256,uint256,uint256) (PancakeRouter01.sol#325-352) ignores return value by IPancakeFactory(factory).createPair(tokenA,tokenB) (PancakeRouter01.sol#335)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

PancakeRouter01.constructor(address,address)._factory (PancakeRouter01.sol#315) lacks a zero-check on :
        - factory = _factory (PancakeRouter01.sol#316)
PancakeRouter01.constructor(address,address)._WETH (PancakeRouter01.sol#315) lacks a zero-check on :
        - WETH = _WETH (PancakeRouter01.sol#317)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

TransferHelper.safeApprove(address,address,uint256) (PancakeRouter01.sol#22-26) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version=0.6.6 (PancakeRouter01.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

Low level call in TransferHelper.safeApprove(address,address,uint256) (PancakeRouter01.sol#22-26):
    - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (PancakeRouter01.sol#24)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (PancakeRouter01.sol#28-32):
    - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (PancakeRouter01.sol#30)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (PancakeRouter01.sol#34-38):
    - (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (PancakeRouter01.sol#36)
Low level call in TransferHelper.safeTransferETH(address,uint256) (PancakeRouter01.sol#40-43):
    - (success) = to.call{value: value}(new bytes(0)) (PancakeRouter01.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IPancakePair.DOMAIN_SEPARATOR() (PancakeRouter01.sol#61) is not in mixedCase
Function IPancakePair.PERMIT_TYPEHASH() (PancakeRouter01.sol#62) is not in mixedCase
Function IPancakePair.MINIMUM_LIQUIDITY() (PancakeRouter01.sol#79) is not in mixedCase
Function IPancakeRouter01.WETH() (PancakeRouter01.sol#192) is not in mixedCase
Variable PancakeRouter01.WETH (PancakeRouter01.sol#308) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (PancakeRouter01.sol#197) is too similar to
IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (PancakeRouter01.sol#198)
Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (PancakeRouter01.sol#197) is too similar to
PancakeRouter01._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (PancakeRouter01.sol#329)
Variable PancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (PancakeRouter01.sol#356) is too similar to
PancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (PancakeRouter01.sol#357)
Variable PancakeRouter01._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (PancakeRouter01.sol#328) is too similar to
PancakeRouter01._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (PancakeRouter01.sol#329)
Variable PancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (PancakeRouter01.sol#356) is too similar to
PancakeRouter01._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (PancakeRouter01.sol#329)
Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (PancakeRouter01.sol#197) is too similar to
PancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (PancakeRouter01.sol#357)
Variable PancakeRouter01._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (PancakeRouter01.sol#328) is too similar to
PancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (PancakeRouter01.sol#357)
Variable PancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (PancakeRouter01.sol#356) is too similar to
IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (PancakeRouter01.sol#198)
Variable PancakeRouter01._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (PancakeRouter01.sol#328) is too similar to
IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (PancakeRouter01.sol#198)
Variable PancakeRouter01._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountAOptimal (PancakeRouter01.sol#346) is too similar to
PancakeRouter01._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBOptimal (PancakeRouter01.sol#341)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

quote(uint256,uint256,uint256) should be declared external:
    - PancakeRouter01.quote(uint256,uint256,uint256) (PancakeRouter01.sol#556-558)
getAmountOut(uint256,uint256,uint256) should be declared external:
```

```
    - PancakeRouter01.getAmountOut(uint256,uint256,uint256) (PancakeRouter01.sol#560-562)
getAmountIn(uint256,uint256,uint256) should be declared external:
    - PancakeRouter01.getAmountIn(uint256,uint256,uint256) (PancakeRouter01.sol#564-566)
getAmountsOut(uint256,address[]) should be declared external:
    - PancakeRouter01.getAmountsOut(uint256,address[]) (PancakeRouter01.sol#568-570)
getAmountsIn(uint256,address[]) should be declared external:
    - PancakeRouter01.getAmountsIn(uint256,address[]) (PancakeRouter01.sol#572-574)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
PancakeRouter01.sol analyzed (9 contracts with 77 detectors), 33 result(s) found


slither PancakeRouter.sol
Compilation warnings/errors on PancakeRouter.sol:
Warning: Contract code size exceeds 24576 bytes (a limit introduced in Spurious Dragon). This contract may not be deployable on mainnet. Consider enabling the optimizer (with a low "runs" value!), turning off revert
strings, or using libraries.
   --> PancakeRouter.sol:347:1:
    |
347 | contract PancakeRouter is IPancakeRouter02 {
    | ^ (Relevant source part starts here and spans across multiple lines).



PancakeRouter.removeLiquidity(address,address,uint256,uint256,uint256,address,uint256) (PancakeRouter.sol#438-454) ignores return value by IPancakePair(pair).transferFrom(msg.sender,pair,liquidity) (PancakeRouter.sol#448)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

PancakeLibrary.getAmountsOut(address,uint256,address[]).i (PancakeRouter.sol#307) is a local variable never initialized
PancakeRouter._swap(uint256[],address[],address).i (PancakeRouter.sol#548) is a local variable never initialized
PancakeRouter._swapSupportingFeeOnTransferTokens(address[],address).i (PancakeRouter.sol#657) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256) (PancakeRouter.sol#368-395) ignores return value by IPancakeFactory(factory).createPair(tokenA,tokenB) (PancakeRouter.sol#378)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

PancakeRouter.constructor(address,address)._factory (PancakeRouter.sol#358) lacks a zero-check on :
        - factory = _factory (PancakeRouter.sol#359)
PancakeRouter.constructor(address,address)._WETH (PancakeRouter.sol#358) lacks a zero-check on :
        - WETH = _WETH (PancakeRouter.sol#360)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

TransferHelper.safeApprove(address,address,uint256) (PancakeRouter.sol#22-26) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version=0.6.6 (PancakeRouter.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

Low level call in TransferHelper.safeApprove(address,address,uint256) (PancakeRouter.sol#22-26):
    - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (PancakeRouter.sol#24)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (PancakeRouter.sol#28-32):
    - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (PancakeRouter.sol#30)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (PancakeRouter.sol#34-38):
    - (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (PancakeRouter.sol#36)
Low level call in TransferHelper.safeTransferETH(address,uint256) (PancakeRouter.sol#40-43):
    - (success) = to.call{value: value}(new bytes(0)) (PancakeRouter.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IPancakeRouter01.WETH() (PancakeRouter.sol#48) is not in mixedCase
Function IPancakePair.DOMAIN_SEPARATOR() (PancakeRouter.sol#196) is not in mixedCase
Function IPancakePair.PERMIT_TYPEHASH() (PancakeRouter.sol#197) is not in mixedCase
Function IPancakePair.MINIMUM_LIQUIDITY() (PancakeRouter.sol#214) is not in mixedCase
Variable PancakeRouter.WETH (PancakeRouter.sol#351) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (PancakeRouter.sol#53) is too similar to
IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (PancakeRouter.sol#54)
Variable PancakeRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (PancakeRouter.sol#399) is too similar to
PancakeRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (PancakeRouter.sol#400)
Variable PancakeRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (PancakeRouter.sol#399) is too similar to
IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (PancakeRouter.sol#54)
Variable PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (PancakeRouter.sol#371) is too similar to
PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (PancakeRouter.sol#372)
Variable PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (PancakeRouter.sol#371) is too similar to
IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (PancakeRouter.sol#54)
Variable PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (PancakeRouter.sol#371) is too similar to
PancakeRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (PancakeRouter.sol#400)
Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (PancakeRouter.sol#53) is too similar to
PancakeRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (PancakeRouter.sol#400)
Variable PancakeRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (PancakeRouter.sol#399) is too similar to
PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (PancakeRouter.sol#372)
Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (PancakeRouter.sol#53) is too similar to
PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (PancakeRouter.sol#372)
Variable PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountAOptimal (PancakeRouter.sol#389) is too similar to
PancakeRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBOptimal (PancakeRouter.sol#384)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

quote(uint256,uint256,uint256) should be declared external:
    - PancakeRouter.quote(uint256,uint256,uint256) (PancakeRouter.sol#738-740)
getAmountOut(uint256,uint256,uint256) should be declared external:
    - PancakeRouter.getAmountOut(uint256,uint256,uint256) (PancakeRouter.sol#742-750)
getAmountIn(uint256,uint256,uint256) should be declared external:
    - PancakeRouter.getAmountIn(uint256,uint256,uint256) (PancakeRouter.sol#752-760)
getAmountsOut(uint256,address[]) should be declared external:
    - PancakeRouter.getAmountsOut(uint256,address[]) (PancakeRouter.sol#762-770)
getAmountsIn(uint256,address[]) should be declared external:
    - PancakeRouter.getAmountsIn(uint256,address[]) (PancakeRouter.sol#772-780)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
PancakeRouter.sol analyzed (10 contracts with 77 detectors), 34 result(s) found


slither PancakeFactory.sol

PancakePair._update(uint256,uint256,uint112,uint112) (PancakeFactory.sol#318-331) uses a weak PRNG: "blockTimestamp = uint32(block.timestamp % 2 ** 32) (PancakeFactory.sol#320)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

PancakePair._safeTransfer(address,address,uint256) (PancakeFactory.sol#289-292) uses a dangerous strict equality:
    - require(bool,string)(success && (data.length == 0 || abi.decode(data,(bool))),Pancake: TRANSFER_FAILED) (PancakeFactory.sol#291)
PancakePair.mint(address) (PancakeFactory.sol#355-376) uses a dangerous strict equality:
    - _totalSupply == 0 (PancakeFactory.sol#364)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in PancakePair.burn(address) (PancakeFactory.sol#379-401):
    External calls:
    - _safeTransfer(_token0,to,amount0) (PancakeFactory.sol#393)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PancakeFactory.sol#290)
    - _safeTransfer(_token1,to,amount1) (PancakeFactory.sol#394)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PancakeFactory.sol#290)
    State variables written after the call(s):
    - _update(balance0,balance1,_reserve0,_reserve1) (PancakeFactory.sol#398)
        - blockTimestampLast = blockTimestamp (PancakeFactory.sol#329)
    - kLast = uint256(reserve0).mul(reserve1) (PancakeFactory.sol#399)
    - _update(balance0,balance1,_reserve0,_reserve1) (PancakeFactory.sol#398)
        - reserve0 = uint112(balance0) (PancakeFactory.sol#327)
    - _update(balance0,balance1,_reserve0,_reserve1) (PancakeFactory.sol#398)
        - reserve1 = uint112(balance1) (PancakeFactory.sol#328)
Reentrancy in PancakeFactory.createPair(address,address) (PancakeFactory.sol#467-482):
    External calls:
    - IPancakePair(pair).initialize(token0,token1) (PancakeFactory.sol#477)
    State variables written after the call(s):
    - getPair[token0][token1] = pair (PancakeFactory.sol#478)
    - getPair[token1][token0] = pair (PancakeFactory.sol#479)
Reentrancy in PancakePair.swap(uint256,uint256,address,bytes) (PancakeFactory.sol#404-432):
    External calls:
    - _safeTransfer(_token0,to,amount0Out) (PancakeFactory.sol#415)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PancakeFactory.sol#290)
    - _safeTransfer(_token1,to,amount1Out) (PancakeFactory.sol#416)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PancakeFactory.sol#290)
    - IPancakeCallee(to).pancakeCall(msg.sender,amount0Out,amount1Out,data) (PancakeFactory.sol#417)
    State variables written after the call(s):
    - _update(balance0,balance1,_reserve0,_reserve1) (PancakeFactory.sol#430)
        - blockTimestampLast = blockTimestamp (PancakeFactory.sol#329)
    - _update(balance0,balance1,_reserve0,_reserve1) (PancakeFactory.sol#430)
        - reserve0 = uint112(balance0) (PancakeFactory.sol#327)
    - _update(balance0,balance1,_reserve0,_reserve1) (PancakeFactory.sol#430)
        - reserve1 = uint112(balance1) (PancakeFactory.sol#328)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

PancakePair.initialize(address,address)._token0 (PancakeFactory.sol#311) lacks a zero-check on :
        - token0 = _token0 (PancakeFactory.sol#313)
PancakePair.initialize(address,address)._token1 (PancakeFactory.sol#311) lacks a zero-check on :
        - token1 = _token1 (PancakeFactory.sol#314)
PancakeFactory.constructor(address)._feeToSetter (PancakeFactory.sol#459) lacks a zero-check on :
        - feeToSetter = _feeToSetter (PancakeFactory.sol#460)
PancakeFactory.setFeeTo(address)._feeTo (PancakeFactory.sol#484) lacks a zero-check on :
        - feeTo = _feeTo (PancakeFactory.sol#486)
PancakeFactory.setFeeToSetter(address)._feeToSetter (PancakeFactory.sol#489) lacks a zero-check on :
        - feeToSetter = _feeToSetter (PancakeFactory.sol#491)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in PancakePair.burn(address) (PancakeFactory.sol#379-401):
    External calls:
    - _safeTransfer(_token0,to,amount0) (PancakeFactory.sol#393)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PancakeFactory.sol#290)
    - _safeTransfer(_token1,to,amount1) (PancakeFactory.sol#394)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PancakeFactory.sol#290)
    State variables written after the call(s):
    - _update(balance0,balance1,_reserve0,_reserve1) (PancakeFactory.sol#398)
```

```
        - price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (PancakeFactory.sol#324)
    - _update(balance0,balance1,_reserve0,_reserve1) (PancakeFactory.sol#398)
        - price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (PancakeFactory.sol#325)
Reentrancy in PancakeFactory.createPair(address,address) (PancakeFactory.sol#467-482):
    External calls:
    - IPancakePair(pair).initialize(token0,token1) (PancakeFactory.sol#477)
    State variables written after the call(s):
    - allPairs.push(pair) (PancakeFactory.sol#480)
Reentrancy in PancakePair.swap(uint256,uint256,address,bytes) (PancakeFactory.sol#404-432):
    External calls:
    - _safeTransfer(_token0,to,amount0Out) (PancakeFactory.sol#415)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PancakeFactory.sol#290)
    - _safeTransfer(_token1,to,amount1Out) (PancakeFactory.sol#416)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PancakeFactory.sol#290)
    - IPancakeCallee(to).pancakeCall(msg.sender,amount0Out,amount1Out,data) (PancakeFactory.sol#417)
    State variables written after the call(s):
    - _update(balance0,balance1,_reserve0,_reserve1) (PancakeFactory.sol#430)
        - price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (PancakeFactory.sol#324)
    - _update(balance0,balance1,_reserve0,_reserve1) (PancakeFactory.sol#430)
        - price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (PancakeFactory.sol#325)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in PancakePair.burn(address) (PancakeFactory.sol#379-401):
    External calls:
    - _safeTransfer(_token0,to,amount0) (PancakeFactory.sol#393)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PancakeFactory.sol#290)
    - _safeTransfer(_token1,to,amount1) (PancakeFactory.sol#394)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PancakeFactory.sol#290)
    Event emitted after the call(s):
    - Burn(msg.sender,amount0,amount1,to) (PancakeFactory.sol#400)
    - Sync(reserve0,reserve1) (PancakeFactory.sol#330)
        - _update(balance0,balance1,_reserve0,_reserve1) (PancakeFactory.sol#398)
Reentrancy in PancakeFactory.createPair(address,address) (PancakeFactory.sol#467-482):
    External calls:
    - IPancakePair(pair).initialize(token0,token1) (PancakeFactory.sol#477)
    Event emitted after the call(s):
    - PairCreated(token0,token1,pair,allPairs.length) (PancakeFactory.sol#481)
Reentrancy in PancakePair.swap(uint256,uint256,address,bytes) (PancakeFactory.sol#404-432):
    External calls:
    - _safeTransfer(_token0,to,amount0Out) (PancakeFactory.sol#415)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PancakeFactory.sol#290)
    - _safeTransfer(_token1,to,amount1Out) (PancakeFactory.sol#416)
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PancakeFactory.sol#290)
    - IPancakeCallee(to).pancakeCall(msg.sender,amount0Out,amount1Out,data) (PancakeFactory.sol#417)
    Event emitted after the call(s):
    - Swap(msg.sender,amount0In,amount1In,amount0Out,amount1Out,to) (PancakeFactory.sol#431)
    - Sync(reserve0,reserve1) (PancakeFactory.sol#330)
        - _update(balance0,balance1,_reserve0,_reserve1) (PancakeFactory.sol#430)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

PancakeERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (PancakeFactory.sol#183-195) uses timestamp for comparisons
    Dangerous comparisons:
    - require(bool,string)(deadline >= block.timestamp,Pancake: EXPIRED) (PancakeFactory.sol#184)
PancakePair._update(uint256,uint256,uint112,uint112) (PancakeFactory.sol#318-331) uses timestamp for comparisons
    Dangerous comparisons:
    - timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0 (PancakeFactory.sol#322)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

PancakeERC20.constructor() (PancakeFactory.sol#126-140) uses assembly
    - INLINE ASM (PancakeFactory.sol#128-130)
PancakeFactory.createPair(address,address) (PancakeFactory.sol#467-482) uses assembly
    - INLINE ASM (PancakeFactory.sol#474-476)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Low level call in PancakePair._safeTransfer(address,address,uint256) (PancakeFactory.sol#289-292):
    - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (PancakeFactory.sol#290)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IPancakePair.DOMAIN_SEPARATOR() (PancakeFactory.sol#35) is not in mixedCase
Function IPancakePair.PERMIT_TYPEHASH() (PancakeFactory.sol#36) is not in mixedCase
Function IPancakePair.MINIMUM_LIQUIDITY() (PancakeFactory.sol#53) is not in mixedCase
Function IPancakeERC20.DOMAIN_SEPARATOR() (PancakeFactory.sol#86) is not in mixedCase
Function IPancakeERC20.PERMIT_TYPEHASH() (PancakeFactory.sol#87) is not in mixedCase
Variable PancakeERC20.DOMAIN_SEPARATOR (PancakeFactory.sol#118) is not in mixedCase
Parameter PancakePair.initialize(address,address)._token0 (PancakeFactory.sol#311) is not in mixedCase
Parameter PancakePair.initialize(address,address)._token1 (PancakeFactory.sol#311) is not in mixedCase
Parameter PancakeFactory.setFeeTo(address)._feeTo (PancakeFactory.sol#484) is not in mixedCase
Parameter PancakeFactory.setFeeToSetter(address)._feeToSetter (PancakeFactory.sol#489) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable PancakePair.swap(uint256,uint256,address,bytes).balance0Adjusted (PancakeFactory.sol#425) is too similar to PancakePair.swap(uint256,uint256,address,bytes).balance1Adjusted (PancakeFactory.sol#426)
Variable PancakePair.price0CumulativeLast (PancakeFactory.sol#271) is too similar to PancakePair.price1CumulativeLast (PancakeFactory.sol#272)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

PancakeFactory.createPair(address,address) (PancakeFactory.sol#467-482) uses literals with too many digits:
    - bytecode = type(address)(PancakePair).creationCode (PancakeFactory.sol#472)
PancakeFactory.slitherConstructorConstantVariables() (PancakeFactory.sol#448-493) uses literals with too many digits:
    - INIT_CODE_PAIR_HASH = keccak256(bytes)(abi.encodePacked(type(address)(PancakePair).creationCode)) (PancakeFactory.sol#449)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
PancakeFactory.sol analyzed (11 contracts with 77 detectors), 36 result(s) found

slither-check-erc PancakeFactory.sol PancakeERC20
# Check PancakeERC20

-## Check functions
[ ✓] totalSupply() is present
    [ ✓] totalSupply() -> () (correct return value)
    [ ✓] totalSupply() is view
[ ✓] balanceOf(address) is present
    [ ✓] balanceOf(address) -> () (correct return value)
    [ ✓] balanceOf(address) is view
[ ✓] transfer(address,uint256) is present
    [ ✓] transfer(address,uint256) -> () (correct return value)
    [ ✓] Transfer(address,address,uint256) is emitted
[ ✓] transferFrom(address,address,uint256) is present
    [ ✓] transferFrom(address,address,uint256) -> () (correct return value)
    [ ✓] Transfer(address,address,uint256) is emitted
[ ✓] approve(address,uint256) is present
    [ ✓] approve(address,uint256) -> () (correct return value)
    [ ✓] Approval(address,address,uint256) is emitted
[ ✓] allowance(address,address) is present
    [ ✓] allowance(address,address) -> () (correct return value)
    [ ✓] allowance(address,address) is view
[ ✓] name() is present
    [ ✓] name() -> () (correct return value)
    [ ✓] name() is view
[ ✓] symbol() is present
    [ ✓] symbol() -> () (correct return value)
    [ ✓] symbol() is view
[ ✓] decimals() is present
    [ ✓] decimals() -> () (correct return value)
    [ ✓] decimals() is view

-## Check events
[ ✓] Transfer(address,address,uint256) is present
    [ ✓] parameter 0 is indexed
    [ ✓] parameter 1 is indexed
[ ✓] Approval(address,address,uint256) is present
    [ ✓] parameter 0 is indexed
    [ ✓] parameter 1 is indexed


# Check PancakePair

-## Check functions
[ ✓] totalSupply() is present
    [ ✓] totalSupply() -> () (correct return value)
    [ ✓] totalSupply() is view
[ ✓] balanceOf(address) is present
    [ ✓] balanceOf(address) -> () (correct return value)
    [ ✓] balanceOf(address) is view
[ ✓] transfer(address,uint256) is present
    [ ✓] transfer(address,uint256) -> () (correct return value)
    [ ✓] Transfer(address,address,uint256) is emitted
[ ✓] transferFrom(address,address,uint256) is present
    [ ✓] transferFrom(address,address,uint256) -> () (correct return value)
    [ ✓] Transfer(address,address,uint256) is emitted
[ ✓] approve(address,uint256) is present
    [ ✓] approve(address,uint256) -> () (correct return value)
    [ ✓] Approval(address,address,uint256) is emitted
[ ✓] allowance(address,address) is present
    [ ✓] allowance(address,address) -> () (correct return value)
    [ ✓] allowance(address,address) is view
[ ✓] name() is present
    [ ✓] name() -> () (correct return value)
    [ ✓] name() is view
[ ✓] symbol() is present
    [ ✓] symbol() -> () (correct return value)
    [ ✓] symbol() is view
[ ✓] decimals() is present
    [ ✓] decimals() -> () (correct return value)
    [ ✓] decimals() is view

-## Check events
[ ✓] Transfer(address,address,uint256) is present
    [ ✓] parameter 0 is indexed
```

```
    [ ✓] parameter 1 is indexed
[ ✓] Approval(address,address,uint256) is present
    [ ✓] parameter 0 is indexed
    [ ✓] parameter 1 is indexed


    [ ] PancakeERC20 is not protected for the ERC20 approval race condition
    [ ] PancakePair is not protected for the ERC20 approval race condition
```

Adherence to Best Practices

1. It is important to validate inputs, even if they only come from trusted addresses, to avoid human error. Arguably the decision to omit non-zero or `address(0)` checks can be made due to gas saving reasons, this decision should however be made consciously or corresponding checks should be added. (**Update**: Unresolved)

2. To facilitate smart contract monitoring by other systems which want to integrate with the smart contract it is recommended to emit events on state changes. Since in `PancakeFactory.setFeeTo()` and `PancakeFactory.setFeeToSetter()` state variables are modified corresponding events should be emitted. (**Update**: Unresolved)

3. To improve readability, auditability and therefore code quality, it is recommended to removed dead code, code that is never executed, or code that does nothing. In file `PancakeRouter01.sol` (and the corresponding inlined copy in file `PancakeRouter.sol`) in `PancakeLibrary.getReserves()` the call to `pairFor(factory, tokenA, tokenB);` does neither change state, nor is the return value used and should therefore be removed. (**Update**: Unresolved)

Test Results

Test Suite Results

The repository did not contain any tests, which could've been run.

Update:
For the re-audit the team has provided tests, based on tests from Uniswap and PancakeSwap, all of which are passing.

```
PancakeERC20
   ✓ name, symbol, decimals, totalSupply, balanceOf, DOMAIN_SEPARATOR, PERMIT_TYPEHASH (161ms)
   ✓ approve (135ms)
   ✓ transfer (142ms)
   ✓ transfer:fail (52ms)
   ✓ transferFrom (212ms)
   ✓ transferFrom:max (160ms)
   ✓ permit (126ms)

PancakeFactory
   ✓ feeTo, feeToSetter, allPairsLength (38ms)
   ✓ createPair (283ms)
   ✓ createPair:reverse (255ms)
   ✓ createPair:gas (107ms)
   ✓ setFeeTo (65ms)
   ✓ setFeeToSetter (80ms)

PancakePair
   ✓ mint (218ms)
   ✓ getInputPrice:0 (268ms)
   ✓ getInputPrice:1 (271ms)
   ✓ getInputPrice:2 (259ms)
   ✓ getInputPrice:3 (267ms)
   ✓ getInputPrice:4 (265ms)
   ✓ getInputPrice:5 (255ms)
   ✓ getInputPrice:6 (264ms)
   ✓ optimistic:0 (261ms)
   ✓ optimistic:1 (239ms)
   ✓ optimistic:2 (242ms)
   ✓ optimistic:3 (242ms)
   ✓ swap:token0 (279ms)
   ✓ swap:token1 (278ms)
   ✓ swap:gas (266ms)
   ✓ burn (289ms)
   ✓ price{0,1}CumulativeLast (343ms)
   ✓ feeTo:off (302ms)
   ✓ feeTo:on (391ms)


  32 passing (9s)


  UniswapV2Router01
    UniswapV2Router02
WARNING: unsupported ABI type - receive
WARNING: unsupported ABI type - receive
     ✓ factory, WETH
     ✓ addLiquidity (279ms)
     ✓ addLiquidityETH (222ms)
     ✓ removeLiquidity (432ms)
     ✓ removeLiquidityETH (409ms)
     ✓ removeLiquidityWithPermit (244ms)
     ✓ removeLiquidityETHWithPermit (316ms)
    swapExactTokensForTokens
       ✓ happy path (75ms)
       ✓ amounts (107ms)
       ✓ gas (158ms)
    swapTokensForExactTokens
       ✓ happy path (108ms)
       ✓ amounts (105ms)
    swapExactETHForTokens
       ✓ happy path (85ms)
       ✓ amounts (65ms)
       ✓ gas (258ms)
    swapTokensForExactETH
       ✓ happy path (120ms)
       ✓ amounts (110ms)
    swapExactTokensForETH
       ✓ happy path (121ms)
       ✓ amounts (102ms)
    swapETHForExactTokens
       ✓ happy path (72ms)
       ✓ amounts (63ms)

  UniswapV2Router02
WARNING: unsupported ABI type - receive
WARNING: unsupported ABI type - receive
     ✓ quote (39ms)
     ✓ getAmountOut
     ✓ getAmountIn
     ✓ getAmountsOut (153ms)
     ✓ getAmountsIn (167ms)

  fee-on-transfer tokens
WARNING: unsupported ABI type - receive
WARNING: unsupported ABI type - receive
     ✓ removeLiquidityETHSupportingFeeOnTransferTokens (259ms)
     ✓ removeLiquidityETHWithPermitSupportingFeeOnTransferTokens (304ms)
     ✓ swapExactETHForTokensSupportingFeeOnTransferTokens (175ms)
     ✓ swapExactTokensForETHSupportingFeeOnTransferTokens (241ms)
    swapExactTokensForTokensSupportingFeeOnTransferTokens
       ✓ DTT -> WETH (101ms)
       ✓ WETH -> DTT (149ms)

  fee-on-transfer tokens: reloaded
    swapExactTokensForTokensSupportingFeeOnTransferTokens
WARNING: unsupported ABI type - receive
WARNING: unsupported ABI type - receive
       ✓ DTT -> DTT2 (115ms)


  33 passing (13s)
```

Code Coverage

The repository did not contain any tests, for which coverages could've been computed.

Update: The newly provided tests were provided without coverage instructions.

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Contracts

```
05d6f9a67f4832304c0713b324b1ffbe9e6d0d907a69671a7370e77334297f13 ./contracts/arthswap-periphery/contracts/PancakeRouter.sol
8f24aabcaea01638f5d4a312b9c8d8ac69cdb15a2d5b4d38556489bb3fc555d5 ./contracts/arthswap-periphery/contracts/test/DeflatingERC20.sol
a466938820fd1779db197c138f53d384e5ab2567b677af2bdb59841eb0f676a3 ./contracts/arthswap-periphery/contracts/test/RouterEventEmitter.sol
4c14359780b17cfc9648925e2c1182e7a64a273d0c71212e187e736fa734744f ./contracts/arthswap-periphery/contracts/test/ERC20.sol
33f13565b207188485030aa20fd1f38d52d8c455cc64720560fb38149dcb4137 ./contracts/arthswap-periphery/contracts/test/WETH9.sol
b34797b7afff45d4de8156ad7daeeb5c99cc62e1c85d561a29168cfbf3125c22 ./contracts/arthswap-core/contracts/PancakeFactory.sol
3f486abbc3ffb935246dcad08dc0b7c5ea708d035556ab7a2e9f62a632456962 ./contracts/arthswap-core/contracts/test/TestPancakeERC20.sol
b41b463b016b64daf736a003c2dd2a3d98fa51e6058bd69bb86ee52bb9e99b33 ./contracts/arthswap-core/contracts/test/ERC20.sol
```

Tests

```
f41c12a97b102e7b6e62e893e6fbeb4b6b7bca7bf2eb3ee9c0f7a3631165b841 ./tests/arthswap-periphery/test/UniswapV2Router01.spec.ts
7cf9af4de63be6a784521ee8d02a586fe17cecc706a5d3bfb1e468e04924654a ./tests/arthswap-periphery/test/UniswapV2Router02.spec.ts
8c3fda14057009d94758a0c98a19497db2ae4141a99a9df7a45039761dca3231 ./tests/arthswap-periphery/test/shared/utilities.ts
29f6a2ec0f5cc04263cbd97185c24482898c6c607d801e32413ef52a5b3cfd0f ./tests/arthswap-periphery/test/shared/fixtures.ts
3912611ec9ffc3f159b38cb5f40ce4e9d16b29f672d06947be2cb7861dc1c1e8 ./tests/arthswap-core/test/PancakeFactory.spec.ts
14550a6d1e59f9c4e7847673930594628024e3e88b4bcfdf00809a7b62a6c109 ./tests/arthswap-core/test/PancakePair.spec.ts
2fce88b35ebabfe3ebbe8021c69a45572bdcab1b74aad2b2ff923bb45274c417 ./tests/arthswap-core/test/PancakeERC20.spec.ts
d082b920be2052397b5e6f0a00d3948b416ecd5a2623353365134456e3cda026 ./tests/arthswap-core/test/shared/utilities.ts
1c572481349ea0865ea99088ceefd3bb4df20ae8f5399d04fe28add171ac0d42 ./tests/arthswap-core/test/shared/fixtures.ts
```

Changelog

- 2022-01-29 - Initial report

About Quantstamp

Quantstamp is a Y Combinator-backed company that helps to secure blockchain platforms at scale using computer-aided reasoning tools, with a mission to help boost the adoption of this exponentially growing technology.

With over 1000 Google scholar citations and numerous published papers, Quantstamp's team has decades of combined experience in formal verification, static analysis, and software verification. Quantstamp has also developed a protocol to help smart contract developers and projects worldwide to perform cost-effective smart contract security scans.

To date, Quantstamp has protected $5B in digital asset risk from hackers and assisted dozens of blockchain projects globally through its white glove security assessment services. As an evangelist of the blockchain ecosystem, Quantstamp assists core infrastructure projects and leading community initiatives such as the Ethereum Community Fund to expedite the adoption of blockchain technology.

Quantstamp's collaborations with leading academic institutions such as the National University of Singapore and MIT (Massachusetts Institute of Technology) reflect our commitment to research, development, and enabling world-class blockchain security.