DESSERT
FINANCE

SmexRouter.sol V2

Fork Validation Report

Completed at block **20964225**

PERFORMED BY DESSERT FINANCE
FOR CONTRACT ADDRESS: PROVIDED SOL FILES

INITIAL DISCLAIMER

Dessert Finance provides due-diligence project audits for various projects. Dessert Finance in no way guarantees that a project will not remove liquidity, sell off team supply, or otherwise exit scam.

Dessert Finance does the legwork and provides public information about the project in an easy-to-understand format for the common person.

Agreeing to an audit in no way guarantees that a team will not remove ***all*** liquidity (“Rug Pull”), remove liquidity slowly, sell off tokens, quit the project, or completely exit scam. There is also no way to prevent private sale holders from selling off their tokens. It is ultimately your responsibility to read through all documentation, social media posts, and contract code of each individual project to draw your own conclusions and set your own risk tolerance.

Dessert Finance in no way takes responsibility for any losses, nor does Dessert Finance encourage any speculative investments. The information provided in this audit is for information purposes only and should not be considered investment advice. Dessert Finance does not endorse, recommend, support, or suggest any projects that have been audited. An audit is an informational report based on our findings, We recommend you do your own research, we will never endorse any project to invest in.

Table of Contents

BSC Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on SmexRouter.sol

Contract Address
N/A

Contract Creator
N/A

Source Code
Not Deployed

Contract Name
SmexRouter

Other Settings
default evmVersion, None

Compiler Version
v0.6.6+commit. 6c089d02

Optimization Enabled
Yes with 200 runs

The contract code is **NOT DEPLOYED** on BSCScan.

Contract Code Audit – Code Comparison

The Following changes were found to the forked contracts

Original Code

833 Lines

Forked Code Changes

95 Changes

Original Compiler

^0.6.6

Forked Code Compiler

^0.6.6

SmexRouter Fork Modifications

The following **2x** modifications were found. Original PancakeSwap V2 Router contracts are represented in blue/dark blue. Modified SmexRouter contracts are represented by purple/dark purple.

The risk of unexpected functionality is Low.

```
// File: contracts\interfaces\IPancakeRouter01.sol

pragma solidity >=0.6.2;

interface IPancakeRouter01 {
    function factory() external pure returns (address);
    function WETH() external pure returns (address);

    function addLiquidity(
        address tokenA,
        address tokenB,
        uint amountADesired,
        uint amountBDesired,
        uint amountAMin,
        uint amountBMin,
        address to,
        uint deadline
    ) external returns (uint amountA, uint amountB, uint liquidity);
```

```
// File: contracts\interfaces\ISmexRouter01.sol

pragma solidity >=0.6.2;

interface ISmexRouter01 {
    function factory() external pure returns (address);
    function WETH() external pure returns (address);

    function addLiquidity(
        address tokenA,
        address tokenB,
        uint amountADesired,
        uint amountBDesired,
        uint amountAMin,
        uint amountBMin,
        address to,
        uint deadline
    ) external returns (uint amountA, uint amountB, uint liquidity);
```

SmexRouter Fork Modifications

The following **2x** modifications were found. Original PancakeSwap V2 Router contracts are represented in blue/dark blue. Modified SmexRouter contracts are represented by purple/dark purple.

The risk of unexpected functionality is Low.

```
// File: contracts\interfaces\IPancakeRouter02.sol

pragma solidity >=0.6.2;

interface IPancakeRouter02 is IPancakeRouter01 {
    function removeLiquidityETHSupportingFeeOnTransferTokens(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
    ) external returns (uint amountETH);
    function removeLiquidityETHWithPermitSupportingFeeOnTransferTok
ens(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
```

```
// File: contracts\interfaces\ISmexRouter02.sol

pragma solidity >=0.6.2;

interface ISmexRouter02 is ISmexRouter01 {
    function removeLiquidityETHSupportingFeeOnTransferTokens(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
    ) external returns (uint amountETH);
    function removeLiquidityETHWithPermitSupportingFeeOnTransferTok
ens(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
```

SmexRouter Fork Modifications

The following **2x** modifications were found. Original PancakeSwap V2 Router contracts are represented in blue/dark blue. Modified SmexRouter contracts are represented by purple/dark purple.

The risk of unexpected functionality is Low.

```
178 // File: contracts\interfaces\IPancakeFactory.sol
179
180 pragma solidity >=0.5.0;
181
182 interface IPancakeFactory {
183     event PairCreated(address indexed token0, address indexed token
    1, address pair, uint);
184
185     function feeTo() external view returns (address);
186     function feeToSetter() external view returns (address);
187
188     function getPair(address tokenA, address tokenB) external view
    returns (address pair);
189     function allPairs(uint) external view returns (address pair);
190     function allPairsLength() external view returns (uint);
191
192     function createPair(address tokenA, address tokenB) external re
    turns (address pair);
193
194     function setFeeTo(address) external;
```

```
178 // File: contracts\interfaces\ISmexFactory.sol
179
180 pragma solidity >=0.5.0;
181
182 interface ISmexFactory {
183     event PairCreated(address indexed token0, address indexed token
    1, address pair, uint);
184
185     function feeTo() external view returns (address);
186     function feeToSetter() external view returns (address);
187
188     function getPair(address tokenA, address tokenB) external view
    returns (address pair);
189     function allPairs(uint) external view returns (address pair);
190     function allPairsLength() external view returns (uint);
191
192     function createPair(address tokenA, address tokenB) external re
    turns (address pair);
193
194     function setFeeTo(address) external;
```

SmexRouter Fork Modifications

The following **2x** modifications were found. Original PancakeSwap V2 Router contracts are represented in blue/dark blue. Modified SmexRouter contracts are represented by purple/dark purple.

The risk of unexpected functionality is Low.

```
218 }
219
220 // File: contracts\interfaces\IPancakePair.sol
221
222 pragma solidity >=0.5.0;
223
224 interface IPancakePair {
225     event Approval(address indexed owner, address indexed spender,
    uint value);
226     event Transfer(address indexed from, address indexed to, uint v
    alue);
227
228     function name() external pure returns (string memory);
229     function symbol() external pure returns (string memory);
230     function decimals() external pure returns (uint8);
231     function totalSupply() external view returns (uint);
232     function balanceOf(address owner) external view returns (uint);
233     function allowance(address owner, address spender) external vie
    w returns (uint);
```

```
218 }
219
220 // File: contracts\interfaces\ISmexPair.sol
221
222 pragma solidity >=0.5.0;
223
224 interface ISmexPair {
225     event Approval(address indexed owner, address indexed spender,
    uint value);
226     event Transfer(address indexed from, address indexed to, uint v
    alue);
227
228     function name() external pure returns (string memory);
229     function symbol() external pure returns (string memory);
230     function decimals() external pure returns (uint8);
231     function totalSupply() external view returns (uint);
232     function balanceOf(address owner) external view returns (uint);
233     function allowance(address owner, address spender) external vie
    w returns (uint);
```

SmexRouter Fork Modifications

The following **6x** modifications were found. Original PancakeSwap V2 Router contracts are represented in blue/dark blue. Modified SmexRouter contracts are represented by purple/dark purple.

The risk of unexpected functionality is Low.

```
// File: contracts\libraries\PancakeLibrary.sol

pragma solidity >=0.5.0;



library PancakeLibrary {
    using SafeMath for uint;

    // returns sorted token addresses, used to handle return values from pairs sorted in this order
    function sortTokens(address tokenA, address tokenB) internal pure returns (address token0, address token1) {
        require(tokenA != tokenB, 'PancakeLibrary: IDENTICAL_ADDRESSES');
        (token0, token1) = tokenA < tokenB ? (tokenA, tokenB) : (tokenB, tokenA);
        require(token0 != address(0), 'PancakeLibrary: ZERO_ADDRESS');
    }

    // calculates the CREATE2 address for a pair without making any external calls
    function pairFor(address factory, address tokenA, address tokenB) internal pure returns (address pair) {
        (address token0, address token1) = sortTokens(tokenA, tokenB);
        pair = address(uint(keccak256(abi.encodePacked(
                hex'ff',
                factory,
                keccak256(abi.encodePacked(token0, token1)),
                hex'00fb7f630766e6a796048ea87d01acd3068e8ff67d078148a3fa3f4a84f69bd5' // init code hash
            ))));
    }

    // fetches and sorts the reserves for a pair
    function getReserves(address factory, address tokenA, address tokenB) internal view returns (uint reserveA, uint reserveB) {
        (address token0,) = sortTokens(tokenA, tokenB);
        pairFor(factory, tokenA, tokenB);
        (uint reserve0, uint reserve1,) = IPancakePair(pairFor(factory, tokenA, tokenB)).getReserves();
        (reserveA, reserveB) = tokenA == token0 ? (reserve0, reserve1) : (reserve1, reserve0);
```

```
// File: contracts\libraries\SmexLibrary.sol

pragma solidity >=0.5.0;



library SmexLibrary {
    using SafeMath for uint;

    // returns sorted token addresses, used to handle return values from pairs sorted in this order
    function sortTokens(address tokenA, address tokenB) internal pure returns (address token0, address token1) {
        require(tokenA != tokenB, 'SmexLibrary: IDENTICAL_ADDRESSES');
        (token0, token1) = tokenA < tokenB ? (tokenA, tokenB) : (tokenB, tokenA);
        require(token0 != address(0), 'SmexLibrary: ZERO_ADDRESS');
    }

    // calculates the CREATE2 address for a pair without making any external calls
    function pairFor(address factory, address tokenA, address tokenB) internal pure returns (address pair) {
        (address token0, address token1) = sortTokens(tokenA, tokenB);
        pair = address(uint(keccak256(abi.encodePacked(
                hex'ff',
                factory,
                keccak256(abi.encodePacked(token0, token1)),
                hex'0406938e4eb92afe1cb10d5fd797c56b2d40da07ad6a1acb826c63ea1e350abd' // init code hash
            ))));
    }

    // fetches and sorts the reserves for a pair
    function getReserves(address factory, address tokenA, address tokenB) internal view returns (uint reserveA, uint reserveB) {
        (address token0,) = sortTokens(tokenA, tokenB);
        pairFor(factory, tokenA, tokenB);
        (uint reserve0, uint reserve1,) = ISmexPair(pairFor(factory, tokenA, tokenB)).getReserves();
        (reserveA, reserveB) = tokenA == token0 ? (reserve0, reserve1) : (reserve1, reserve0);
```

SmexRouter Fork Modifications

The following **6x** modifications were found. Original PancakeSwap V2 Router contracts are represented in blue/dark blue. Modified SmexRouter contracts are represented by purple/dark purple.

The risk of unexpected functionality is Low.

```
312         require(amountA > 0, 'PancakeLibrary: INSUFFICIENT_AMOUNT');
313         require(reserveA > 0 && reserveB > 0, 'PancakeLibrary: INSUFFICIENT_LIQUIDITY');
314         amountB = amountA.mul(reserveB) / reserveA;
315     }
316
317     // given an input amount of an asset and pair reserves, returns the maximum output amount of the other asset
318     function getAmountOut(uint amountIn, uint reserveIn, uint reserveOut) internal pure returns (uint amountOut) {
319         require(amountIn > 0, 'PancakeLibrary: INSUFFICIENT_INPUT_AMOUNT');
320         require(reserveIn > 0 && reserveOut > 0, 'PancakeLibrary: INSUFFICIENT_LIQUIDITY');
321         uint amountInWithFee = amountIn.mul(9975);
322         uint numerator = amountInWithFee.mul(reserveOut);
323         uint denominator = reserveIn.mul(10000).add(amountInWithFee);
324         amountOut = numerator / denominator;
325     }
326
327     // given an output amount of an asset and pair reserves, returns a required input amount of the other asset
328     function getAmountIn(uint amountOut, uint reserveIn, uint reserveOut) internal pure returns (uint amountIn) {
329         require(amountOut > 0, 'PancakeLibrary: INSUFFICIENT_OUTPUT_AMOUNT');
330         require(reserveIn > 0 && reserveOut > 0, 'PancakeLibrary: INSUFFICIENT_LIQUIDITY');
331         uint numerator = reserveIn.mul(amountOut).mul(10000);
332         uint denominator = reserveOut.sub(amountOut).mul(9975);
333         amountIn = (numerator / denominator).add(1);
```

```
312         require(amountA > 0, 'SmexLibrary: INSUFFICIENT_AMOUNT');
313         require(reserveA > 0 && reserveB > 0, 'SmexLibrary: INSUFFICIENT_LIQUIDITY');
314         amountB = amountA.mul(reserveB) / reserveA;
315     }
316
317     // given an input amount of an asset and pair reserves, returns the maximum output amount of the other asset
318     function getAmountOut(uint amountIn, uint reserveIn, uint reserveOut) internal pure returns (uint amountOut) {
319         require(amountIn > 0, 'SmexLibrary: INSUFFICIENT_INPUT_AMOUNT');
320         require(reserveIn > 0 && reserveOut > 0, 'SmexLibrary: INSUFFICIENT_LIQUIDITY');
321         uint amountInWithFee = amountIn.mul(9975);
322         uint numerator = amountInWithFee.mul(reserveOut);
323         uint denominator = reserveIn.mul(10000).add(amountInWithFee);
324         amountOut = numerator / denominator;
325     }
326
327     // given an output amount of an asset and pair reserves, returns a required input amount of the other asset
328     function getAmountIn(uint amountOut, uint reserveIn, uint reserveOut) internal pure returns (uint amountIn) {
329         require(amountOut > 0, 'SmexLibrary: INSUFFICIENT_OUTPUT_AMOUNT');
330         require(reserveIn > 0 && reserveOut > 0, 'SmexLibrary: INSUFFICIENT_LIQUIDITY');
331         uint numerator = reserveIn.mul(amountOut).mul(10000);
332         uint denominator = reserveOut.sub(amountOut).mul(9975);
333         amountIn = (numerator / denominator).add(1);
```

SmexRouter Fork Modifications

The following **2x** modifications were found. Original PancakeSwap V2 Router contracts are represented in blue/dark blue. Modified SmexRouter contracts are represented by purple/dark purple.

The risk of unexpected functionality is Low.

```
338         require(path.length >= 2, 'PancakeLibrary: INVALID_PATH');
339         amounts = new uint[](path.length);
340         amounts[0] = amountIn;
341         for (uint i; i < path.length - 1; i++) {
342             (uint reserveIn, uint reserveOut) = getReserves(factor
    y, path[i], path[i + 1]);
343             amounts[i + 1] = getAmountOut(amounts[i], reserveIn, re
    serveOut);
344         }
345     }
346
347     // performs chained getAmountIn calculations on any number of p
    airs
348     function getAmountsIn(address factory, uint amountOut, address
    [] memory path) internal view returns (uint[] memory amounts) {
349         require(path.length >= 2, 'PancakeLibrary: INVALID_PATH');
350         amounts = new uint[](path.length);
351         amounts[amounts.length - 1] = amountOut;
352         for (uint i = path.length - 1; i > 0; i--) {
353             (uint reserveIn, uint reserveOut) = getReserves(factor
    y, path[i - 1], path[i]);
```

```
338         require(path.length >= 2, 'SmexLibrary: INVALID_PATH');
339         amounts = new uint[](path.length);
340         amounts[0] = amountIn;
341         for (uint i; i < path.length - 1; i++) {
342             (uint reserveIn, uint reserveOut) = getReserves(factor
    y, path[i], path[i + 1]);
343             amounts[i + 1] = getAmountOut(amounts[i], reserveIn, re
    serveOut);
344         }
345     }
346
347     // performs chained getAmountIn calculations on any number of p
    airs
348     function getAmountsIn(address factory, uint amountOut, address
    [] memory path) internal view returns (uint[] memory amounts) {
349         require(path.length >= 2, 'SmexLibrary: INVALID_PATH');
350         amounts = new uint[](path.length);
351         amounts[amounts.length - 1] = amountOut;
352         for (uint i = path.length - 1; i > 0; i--) {
353             (uint reserveIn, uint reserveOut) = getReserves(factor
    y, path[i - 1], path[i]);
```

SmexRouter Fork Modifications

The following **4x** modifications were found. Original PancakeSwap V2 Router contracts are represented in blue/dark blue. Modified SmexRouter contracts are represented by purple/dark purple.

The risk of unexpected functionality is Low.

```
388
389 // File: contracts\PancakeRouter.sol
390
391 pragma solidity =0.6.6;
392
393
394
395
396
397
398
399 contract PancakeRouter is IPancakeRouter02 {
400     using SafeMath for uint;
401
402     address public immutable override factory;
403     address public immutable override WETH;
404
405     modifier ensure(uint deadline) {
406         require(deadline >= block.timestamp, 'PancakeRouter: EXPIRE
    D');
407         _;
408     }
409
410     constructor(address _factory, address _WETH) public {
411         factory = _factory;
412         WETH = _WETH;
413     }
414
```

```
388
389 // File: contracts\SmexRouter.sol
390
391 pragma solidity =0.6.6;
392
393
394
395
396
397
398
399 contract SmexRouter is ISmexRouter02 {
400     using SafeMath for uint;
401
402     address public immutable override factory;
403     address public immutable override WETH;
404
405     modifier ensure(uint deadline) {
406         require(deadline >= block.timestamp, 'SmexRouter: EXPIRE
    D');
407         _;
408     }
409
410     constructor(address _factory, address _WETH) public {
411         factory = _factory;
412         WETH = _WETH;
413     }
414
```

SmexRouter Fork Modifications

The following **7x** modifications were found. Original PancakeSwap V2 Router contracts are represented in blue/dark blue. Modified SmexRouter contracts are represented by purple/dark purple.

The risk of unexpected functionality is Low.

```
        // create the pair if it doesn't exist yet
        if (IPancakeFactory(factory).getPair(tokenA, tokenB) == address(0)) {
            IPancakeFactory(factory).createPair(tokenA, tokenB);
        }
        (uint reserveA, uint reserveB) = PancakeLibrary.getReserves(factory, tokenA, tokenB);
        if (reserveA == 0 && reserveB == 0) {
            (amountA, amountB) = (amountADesired, amountBDesired);
        } else {
            uint amountBOptimal = PancakeLibrary.quote(amountADesired, reserveA, reserveB);
            if (amountBOptimal <= amountBDesired) {
                require(amountBOptimal >= amountBMin, 'PancakeRouter: INSUFFICIENT_B_AMOUNT');
                (amountA, amountB) = (amountADesired, amountBOptimal);
            } else {
                uint amountAOptimal = PancakeLibrary.quote(amountBDesired, reserveB, reserveA);
                assert(amountAOptimal <= amountADesired);
                require(amountAOptimal >= amountAMin, 'PancakeRouter: INSUFFICIENT_A_AMOUNT');
                (amountA, amountB) = (amountAOptimal, amountBDesired);
            }
        }
    }
    function addLiquidity(
```

```
        // create the pair if it doesn't exist yet
        if (ISmexFactory(factory).getPair(tokenA, tokenB) == address(0)) {
            ISmexFactory(factory).createPair(tokenA, tokenB);
        }
        (uint reserveA, uint reserveB) = SmexLibrary.getReserves(factory, tokenA, tokenB);
        if (reserveA == 0 && reserveB == 0) {
            (amountA, amountB) = (amountADesired, amountBDesired);
        } else {
            uint amountBOptimal = SmexLibrary.quote(amountADesired, reserveA, reserveB);
            if (amountBOptimal <= amountBDesired) {
                require(amountBOptimal >= amountBMin, 'SmexRouter: INSUFFICIENT_B_AMOUNT');
                (amountA, amountB) = (amountADesired, amountBOptimal);
            } else {
                uint amountAOptimal = SmexLibrary.quote(amountBDesired, reserveB, reserveA);
                assert(amountAOptimal <= amountADesired);
                require(amountAOptimal >= amountAMin, 'SmexRouter: INSUFFICIENT_A_AMOUNT');
                (amountA, amountB) = (amountAOptimal, amountBDesired);
            }
        }
    }
    function addLiquidity(
```

SmexRouter Fork Modifications

The following **4x** modifications were found. Original PancakeSwap V2 Router contracts are represented in blue/dark blue. Modified SmexRouter contracts are represented by purple/dark purple.

The risk of unexpected functionality is Low.

```
458         (amountA, amountB) = _addLiquidity(tokenA, tokenB, amountADesired, amountBDesired, amountAMin, amountBMin);
459         address pair = PancakeLibrary.pairFor(factory, tokenA, tokenB);
460         TransferHelper.safeTransferFrom(tokenA, msg.sender, pair, amountA);
461         TransferHelper.safeTransferFrom(tokenB, msg.sender, pair, amountB);
462         liquidity = IPancakePair(pair).mint(to);
463     }
464     function addLiquidityETH(
465         address token,
466         uint amountTokenDesired,
467         uint amountTokenMin,
468         uint amountETHMin,
469         address to,
470         uint deadline
471     ) external virtual override payable ensure(deadline) returns (uint amountToken, uint amountETH, uint liquidity) {
472         (amountToken, amountETH) = _addLiquidity(
473             token,
474             WETH,
475             amountTokenDesired,
476             msg.value,
477             amountTokenMin,
478             amountETHMin
479         );
480         address pair = PancakeLibrary.pairFor(factory, token, WETH);
481         TransferHelper.safeTransferFrom(token, msg.sender, pair, amountToken);
482         IWETH(WETH).deposit{value: amountETH}();
483         assert(IWETH(WETH).transfer(pair, amountETH));
484         liquidity = IPancakePair(pair).mint(to);
485         // refund dust eth, if any
486         if (msg.value > amountETH) TransferHelper.safeTransferETH(msg.sender, msg.value - amountETH);
```

```
458         (amountA, amountB) = _addLiquidity(tokenA, tokenB, amountADesired, amountBDesired, amountAMin, amountBMin);
459         address pair = SmexLibrary.pairFor(factory, tokenA, tokenB);
460         TransferHelper.safeTransferFrom(tokenA, msg.sender, pair, amountA);
461         TransferHelper.safeTransferFrom(tokenB, msg.sender, pair, amountB);
462         liquidity = ISmexPair(pair).mint(to);
463     }
464     function addLiquidityETH(
465         address token,
466         uint amountTokenDesired,
467         uint amountTokenMin,
468         uint amountETHMin,
469         address to,
470         uint deadline
471     ) external virtual override payable ensure(deadline) returns (uint amountToken, uint amountETH, uint liquidity) {
472         (amountToken, amountETH) = _addLiquidity(
473             token,
474             WETH,
475             amountTokenDesired,
476             msg.value,
477             amountTokenMin,
478             amountETHMin
479         );
480         address pair = SmexLibrary.pairFor(factory, token, WETH);
481         TransferHelper.safeTransferFrom(token, msg.sender, pair, amountToken);
482         IWETH(WETH).deposit{value: amountETH}();
483         assert(IWETH(WETH).transfer(pair, amountETH));
484         liquidity = ISmexPair(pair).mint(to);
485         // refund dust eth, if any
486         if (msg.value > amountETH) TransferHelper.safeTransferETH(msg.sender, msg.value - amountETH);
```

SmexRouter Fork Modifications

The following **6x** modifications were found. Original PancakeSwap V2 Router contracts are represented in blue/dark blue. Modified SmexRouter contracts are represented by purple/dark purple.

The risk of unexpected functionality is Low.

```
A, uint amountB) {
499         address pair = PancakeLibrary.pairFor(factory, tokenA, toke
nB);
500         IPancakePair(pair).transferFrom(msg.sender, pair, liquidit
y); // send liquidity to pair
501         (uint amount0, uint amount1) = IPancakePair(pair).burn(to);
502         (address token0,) = PancakeLibrary.sortTokens(tokenA, token
B);
503         (amountA, amountB) = tokenA == token0 ? (amount0, amount1)
: (amount1, amount0);
504         require(amountA >= amountAMin, 'PancakeRouter: INSUFFICIENT
_A_AMOUNT');
505         require(amountB >= amountBMin, 'PancakeRouter: INSUFFICIENT
_B_AMOUNT');
506     }
507     function removeLiquidityETH(
508         address token,
509         uint liquidity,
510         uint amountTokenMin,
511         uint amountETHMin,
512         address to,
```

```
A, uint amountB) {
499         address pair = SmexLibrary.pairFor(factory, tokenA, token
B);
500         ISmexPair(pair).transferFrom(msg.sender, pair, liquidity);
// send liquidity to pair
501         (uint amount0, uint amount1) = ISmexPair(pair).burn(to);
502         (address token0,) = SmexLibrary.sortTokens(tokenA, tokenB);
503         (amountA, amountB) = tokenA == token0 ? (amount0, amount1)
: (amount1, amount0);
504         require(amountA >= amountAMin, 'SmexRouter: INSUFFICIENT_A_
AMOUNT');
505         require(amountB >= amountBMin, 'SmexRouter: INSUFFICIENT_B_
AMOUNT');
506     }
507     function removeLiquidityETH(
508         address token,
509         uint liquidity,
510         uint amountTokenMin,
511         uint amountETHMin,
512         address to,
```

SmexRouter Fork Modifications

The following **4x** modifications were found. Original PancakeSwap V2 Router contracts are represented in blue/dark blue. Modified SmexRouter contracts are represented by purple/dark purple.

The risk of unexpected functionality is Low.

```
537     ) external virtual override returns (uint amountA, uint amount
    B) {
538         address pair = PancakeLibrary.pairFor(factory, tokenA, toke
    nB);
539         uint value = approveMax ? uint(-1) : liquidity;
540         IPancakePair(pair).permit(msg.sender, address(this), value,
    deadline, v, r, s);
541         (amountA, amountB) = removeLiquidity(tokenA, tokenB, liquid
    ity, amountAMin, amountBMin, to, deadline);
542     }
543     function removeLiquidityETHWithPermit(
544         address token,
545         uint liquidity,
546         uint amountTokenMin,
547         uint amountETHMin,
548         address to,
549         uint deadline,
550         bool approveMax, uint8 v, bytes32 r, bytes32 s
551     ) external virtual override returns (uint amountToken, uint amo
    untETH) {
552         address pair = PancakeLibrary.pairFor(factory, token, WET
    H);
553         uint value = approveMax ? uint(-1) : liquidity;
554         IPancakePair(pair).permit(msg.sender, address(this), value,
    deadline, v, r, s);
555         (amountToken, amountETH) = removeLiquidityETH(token, liquid
    ity, amountTokenMin, amountETHMin, to, deadline);
556     }
557
```

```
537     ) external virtual override returns (uint amountA, uint amount
    B) {
538         address pair = SmexLibrary.pairFor(factory, tokenA, token
    B);
539         uint value = approveMax ? uint(-1) : liquidity;
540         ISmexPair(pair).permit(msg.sender, address(this), value, de
    adline, v, r, s);
541         (amountA, amountB) = removeLiquidity(tokenA, tokenB, liquid
    ity, amountAMin, amountBMin, to, deadline);
542     }
543     function removeLiquidityETHWithPermit(
544         address token,
545         uint liquidity,
546         uint amountTokenMin,
547         uint amountETHMin,
548         address to,
549         uint deadline,
550         bool approveMax, uint8 v, bytes32 r, bytes32 s
551     ) external virtual override returns (uint amountToken, uint amo
    untETH) {
552         address pair = SmexLibrary.pairFor(factory, token, WETH);
553         uint value = approveMax ? uint(-1) : liquidity;
554         ISmexPair(pair).permit(msg.sender, address(this), value, de
    adline, v, r, s);
555         (amountToken, amountETH) = removeLiquidityETH(token, liquid
    ity, amountTokenMin, amountETHMin, to, deadline);
556     }
557
```

SmexRouter Fork Modifications

The following **5x** modifications were found. Original PancakeSwap V2 Router contracts are represented in blue/dark blue. Modified SmexRouter contracts are represented by purple/dark purple.

The risk of unexpected functionality is Low.

```
588     ) external virtual override returns (uint amountETH) {
589         address pair = PancakeLibrary.pairFor(factory, token, WET
    H);
590         uint value = approveMax ? uint(-1) : liquidity;
591         IPancakePair(pair).permit(msg.sender, address(this), value,
    deadline, v, r, s);
592         amountETH = removeLiquidityETHSupportingFeeOnTransferTokens
    (
593             token, liquidity, amountTokenMin, amountETHMin, to, dea
    dline
594         );
595     }
596
597     // **** SWAP ****
598     // requires the initial amount to have already been sent to the
    first pair
599     function _swap(uint[] memory amounts, address[] memory path, ad
    dress _to) internal virtual {
600         for (uint i; i < path.length - 1; i++) {
601             (address input, address output) = (path[i], path[i +
    1]);
602             (address token0,) = PancakeLibrary.sortTokens(input, ou
    tput);
603             uint amountOut = amounts[i + 1];
604             (uint amount0Out, uint amount1Out) = input == token0 ?
    (uint(0), amountOut) : (amountOut, uint(0));
605             address to = i < path.length - 2 ? PancakeLibrary.pairF
    or(factory, output, path[i + 2]) : _to;
606             IPancakePair(PancakeLibrary.pairFor(factory, input, out
    put)).swap(
607                 amount0Out, amount1Out, to, new bytes(0)
608             );
```

```
588     ) external virtual override returns (uint amountETH) {
589         address pair = SmexLibrary.pairFor(factory, token, WETH);
590         uint value = approveMax ? uint(-1) : liquidity;
591         ISmexPair(pair).permit(msg.sender, address(this), value, de
    adline, v, r, s);
592         amountETH = removeLiquidityETHSupportingFeeOnTransferTokens
    (
593             token, liquidity, amountTokenMin, amountETHMin, to, dea
    dline
594         );
595     }
596
597     // **** SWAP ****
598     // requires the initial amount to have already been sent to the
    first pair
599     function _swap(uint[] memory amounts, address[] memory path, ad
    dress _to) internal virtual {
600         for (uint i; i < path.length - 1; i++) {
601             (address input, address output) = (path[i], path[i +
    1]);
602             (address token0,) = SmexLibrary.sortTokens(input, outpu
    t);
603             uint amountOut = amounts[i + 1];
604             (uint amount0Out, uint amount1Out) = input == token0 ?
    (uint(0), amountOut) : (amountOut, uint(0));
605             address to = i < path.length - 2 ? SmexLibrary.pairFor
    (factory, output, path[i + 2]) : _to;
606             ISmexPair(SmexLibrary.pairFor(factory, input, output)).
    swap(
607                 amount0Out, amount1Out, to, new bytes(0)
608             );
```

SmexRouter Fork Modifications

The following **6x** modifications were found. Original PancakeSwap V2 Router contracts are represented in blue/dark blue. Modified SmexRouter contracts are represented by purple/dark purple.

The risk of unexpected functionality is Low.

```
617     ) external virtual override ensure(deadline) returns (uint[] me
    mory amounts) {
618         amounts = PancakeLibrary.getAmountsOut(factory, amountIn, p
    ath);
619         require(amounts[amounts.length - 1] >= amountOutMin, 'Panca
    keRouter: INSUFFICIENT_OUTPUT_AMOUNT');
620         TransferHelper.safeTransferFrom(
621             path[0], msg.sender, PancakeLibrary.pairFor(factory, pa
    th[0], path[1]), amounts[0]
622         );
623         _swap(amounts, path, to);
624     }
625     function swapTokensForExactTokens(
626         uint amountOut,
627         uint amountInMax,
628         address[] calldata path,
629         address to,
630         uint deadline
631     ) external virtual override ensure(deadline) returns (uint[] me
    mory amounts) {
632         amounts = PancakeLibrary.getAmountsIn(factory, amountOut, p
    ath);
633         require(amounts[0] <= amountInMax, 'PancakeRouter: EXCESSIV
    E_INPUT_AMOUNT');
634         TransferHelper.safeTransferFrom(
635             path[0], msg.sender, PancakeLibrary.pairFor(factory, pa
    th[0], path[1]), amounts[0]
636         );
637         _swap(amounts, path, to);
638     }
639     function swapExactETHForTokens(uint amountOutMin, address[] cal
```

```
617     ) external virtual override ensure(deadline) returns (uint[] me
    mory amounts) {
618         amounts = SmexLibrary.getAmountsOut(factory, amountIn, pat
    h);
619         require(amounts[amounts.length - 1] >= amountOutMin, 'SmexR
    outer: INSUFFICIENT_OUTPUT_AMOUNT');
620         TransferHelper.safeTransferFrom(
621             path[0], msg.sender, SmexLibrary.pairFor(factory, path
    [0], path[1]), amounts[0]
622         );
623         _swap(amounts, path, to);
624     }
625     function swapTokensForExactTokens(
626         uint amountOut,
627         uint amountInMax,
628         address[] calldata path,
629         address to,
630         uint deadline
631     ) external virtual override ensure(deadline) returns (uint[] me
    mory amounts) {
632         amounts = SmexLibrary.getAmountsIn(factory, amountOut, pat
    h);
633         require(amounts[0] <= amountInMax, 'SmexRouter: EXCESSIVE_I
    NPUT_AMOUNT');
634         TransferHelper.safeTransferFrom(
635             path[0], msg.sender, SmexLibrary.pairFor(factory, path
    [0], path[1]), amounts[0]
636         );
637         _swap(amounts, path, to);
638     }
639     function swapExactETHForTokens(uint amountOutMin, address[] cal
```

SmexRouter Fork Modifications

The following **8x** modifications were found. Original PancakeSwap V2 Router contracts are represented in blue/dark blue. Modified SmexRouter contracts are represented by purple/dark purple.

The risk of unexpected functionality is Low.

```
        returns (uint[] memory amounts)
    {
        require(path[0] == WETH, 'PancakeRouter: INVALID_PATH');
        amounts = PancakeLibrary.getAmountsOut(factory, msg.value, path);
        require(amounts[amounts.length - 1] >= amountOutMin, 'PancakeRouter: INSUFFICIENT_OUTPUT_AMOUNT');
        IWETH(WETH).deposit{value: amounts[0]}();
        assert(IWETH(WETH).transfer(PancakeLibrary.pairFor(factory, path[0], path[1]), amounts[0]));
        _swap(amounts, path, to);
    }
    function swapTokensForExactETH(uint amountOut, uint amountInMax, address[] calldata path, address to, uint deadline)
        external
        virtual
        override
        ensure(deadline)
        returns (uint[] memory amounts)
    {
        require(path[path.length - 1] == WETH, 'PancakeRouter: INVALID_PATH');
        amounts = PancakeLibrary.getAmountsIn(factory, amountOut, path);
        require(amounts[0] <= amountInMax, 'PancakeRouter: EXCESSIVE_INPUT_AMOUNT');
        TransferHelper.safeTransferFrom(
            path[0], msg.sender, PancakeLibrary.pairFor(factory, path[0], path[1]), amounts[0]
        );
        _swap(amounts, path, address(this));
```

```
        returns (uint[] memory amounts)
    {
        require(path[0] == WETH, 'SmexRouter: INVALID_PATH');
        amounts = SmexLibrary.getAmountsOut(factory, msg.value, path);
        require(amounts[amounts.length - 1] >= amountOutMin, 'SmexRouter: INSUFFICIENT_OUTPUT_AMOUNT');
        IWETH(WETH).deposit{value: amounts[0]}();
        assert(IWETH(WETH).transfer(SmexLibrary.pairFor(factory, path[0], path[1]), amounts[0]));
        _swap(amounts, path, to);
    }
    function swapTokensForExactETH(uint amountOut, uint amountInMax, address[] calldata path, address to, uint deadline)
        external
        virtual
        override
        ensure(deadline)
        returns (uint[] memory amounts)
    {
        require(path[path.length - 1] == WETH, 'SmexRouter: INVALID_PATH');
        amounts = SmexLibrary.getAmountsIn(factory, amountOut, path);
        require(amounts[0] <= amountInMax, 'SmexRouter: EXCESSIVE_INPUT_AMOUNT');
        TransferHelper.safeTransferFrom(
            path[0], msg.sender, SmexLibrary.pairFor(factory, path[0], path[1]), amounts[0]
        );
        _swap(amounts, path, address(this));
```

SmexRouter Fork Modifications

The following **8x** modifications were found. Original PancakeSwap V2 Router contracts are represented in blue/dark blue. Modified SmexRouter contracts are represented by purple/dark purple.

The risk of unexpected functionality is Low.

```
678         require(path[path.length - 1] == WETH, 'PancakeRouter: INVALID_PATH');
679         amounts = PancakeLibrary.getAmountsOut(factory, amountIn, path);
680         require(amounts[amounts.length - 1] >= amountOutMin, 'PancakeRouter: INSUFFICIENT_OUTPUT_AMOUNT');
681         TransferHelper.safeTransferFrom(
682             path[0], msg.sender, PancakeLibrary.pairFor(factory, path[0], path[1]), amounts[0]
683         );
684         _swap(amounts, path, address(this));
685         IWETH(WETH).withdraw(amounts[amounts.length - 1]);
686         TransferHelper.safeTransferETH(to, amounts[amounts.length - 1]);
687     }
688     function swapETHForExactTokens(uint amountOut, address[] calldata path, address to, uint deadline)
689         external
690         virtual
691         override
692         payable
693         ensure(deadline)
694         returns (uint[] memory amounts)
695     {
696         require(path[0] == WETH, 'PancakeRouter: INVALID_PATH');
697         amounts = PancakeLibrary.getAmountsIn(factory, amountOut, path);
698         require(amounts[0] <= msg.value, 'PancakeRouter: EXCESSIVE_INPUT_AMOUNT');
699         IWETH(WETH).deposit{value: amounts[0]}();
700         assert(IWETH(WETH).transfer(PancakeLibrary.pairFor(factory, path[0], path[1]), amounts[0]));
701         _swap(amounts, path, to);
702         // refund dust eth, if any
```

```
678         require(path[path.length - 1] == WETH, 'SmexRouter: INVALID_PATH');
679         amounts = SmexLibrary.getAmountsOut(factory, amountIn, path);
680         require(amounts[amounts.length - 1] >= amountOutMin, 'SmexRouter: INSUFFICIENT_OUTPUT_AMOUNT');
681         TransferHelper.safeTransferFrom(
682             path[0], msg.sender, SmexLibrary.pairFor(factory, path[0], path[1]), amounts[0]
683         );
684         _swap(amounts, path, address(this));
685         IWETH(WETH).withdraw(amounts[amounts.length - 1]);
686         TransferHelper.safeTransferETH(to, amounts[amounts.length - 1]);
687     }
688     function swapETHForExactTokens(uint amountOut, address[] calldata path, address to, uint deadline)
689         external
690         virtual
691         override
692         payable
693         ensure(deadline)
694         returns (uint[] memory amounts)
695     {
696         require(path[0] == WETH, 'SmexRouter: INVALID_PATH');
697         amounts = SmexLibrary.getAmountsIn(factory, amountOut, path);
698         require(amounts[0] <= msg.value, 'SmexRouter: EXCESSIVE_INPUT_AMOUNT');
699         IWETH(WETH).deposit{value: amounts[0]}();
700         assert(IWETH(WETH).transfer(SmexLibrary.pairFor(factory, path[0], path[1]), amounts[0]));
701         _swap(amounts, path, to);
702         // refund dust eth, if any
```

SmexRouter Fork Modifications

The following **5x** modifications were found. Original PancakeSwap V2 Router contracts are represented in blue/dark blue. Modified SmexRouter contracts are represented by purple/dark purple.

The risk of unexpected functionality is Low.

```
1]);
711             (address token0,) = PancakeLibrary.sortTokens(input, ou
tput);
712             IPancakePair pair = IPancakePair(PancakeLibrary.pairFor
(factory, input, output));
713             uint amountInput;
714             uint amountOutput;
715             { // scope to avoid stack too deep errors
716             (uint reserve0, uint reserve1,) = pair.getReserves();
717             (uint reserveInput, uint reserveOutput) = input == toke
n0 ? (reserve0, reserve1) : (reserve1, reserve0);
718             amountInput = IERC20(input).balanceOf(address(pair)).su
b(reserveInput);
719             amountOutput = PancakeLibrary.getAmountOut(amountInput,
reserveInput, reserveOutput);
720             }
721             (uint amount0Out, uint amount1Out) = input == token0 ?
(uint(0), amountOutput) : (amountOutput, uint(0));
722             address to = i < path.length - 2 ? PancakeLibrary.pairF
or(factory, output, path[i + 2]) : _to;
723             pair.swap(amount0Out, amount1Out, to, new bytes(0));
724         }
725     }
```

```
1]);
711             (address token0,) = SmexLibrary.sortTokens(input, outpu
t);
712             ISmexPair pair = ISmexPair(SmexLibrary.pairFor(factory,
input, output));
713             uint amountInput;
714             uint amountOutput;
715             { // scope to avoid stack too deep errors
716             (uint reserve0, uint reserve1,) = pair.getReserves();
717             (uint reserveInput, uint reserveOutput) = input == toke
n0 ? (reserve0, reserve1) : (reserve1, reserve0);
718             amountInput = IERC20(input).balanceOf(address(pair)).su
b(reserveInput);
719             amountOutput = SmexLibrary.getAmountOut(amountInput, re
serveInput, reserveOutput);
720             }
721             (uint amount0Out, uint amount1Out) = input == token0 ?
(uint(0), amountOutput) : (amountOutput, uint(0));
722             address to = i < path.length - 2 ? SmexLibrary.pairFor
(factory, output, path[i + 2]) : _to;
723             pair.swap(amount0Out, amount1Out, to, new bytes(0));
724         }
725     }
```

SmexRouter Fork Modifications

The following **2x** modifications were found. Original PancakeSwap V2 Router contracts are represented in blue/dark blue. Modified SmexRouter contracts are represented by purple/dark purple.

The risk of unexpected functionality is Low.

```
732     ) external virtual override ensure(deadline) {
733         TransferHelper.safeTransferFrom(
734             path[0], msg.sender, PancakeLibrary.pairFor(factory, pa
    th[0], path[1]), amountIn
735         );
736         uint balanceBefore = IERC20(path[path.length - 1]).balanceO
    f(to);
737         _swapSupportingFeeOnTransferTokens(path, to);
738         require(
739             IERC20(path[path.length - 1]).balanceOf(to).sub(balance
    Before) >= amountOutMin,
740             'PancakeRouter: INSUFFICIENT_OUTPUT_AMOUNT'
741         );
742     }
743     function swapExactETHForTokensSupportingFeeOnTransferTokens(
744         uint amountOutMin,
745         address[] calldata path,
746         address to,
747         uint deadline
748     )
```

```
732     ) external virtual override ensure(deadline) {
733         TransferHelper.safeTransferFrom(
734             path[0], msg.sender, SmexLibrary.pairFor(factory, path
    [0], path[1]), amountIn
735         );
736         uint balanceBefore = IERC20(path[path.length - 1]).balanceO
    f(to);
737         _swapSupportingFeeOnTransferTokens(path, to);
738         require(
739             IERC20(path[path.length - 1]).balanceOf(to).sub(balance
    Before) >= amountOutMin,
740             'SmexRouter: INSUFFICIENT_OUTPUT_AMOUNT'
741         );
742     }
743     function swapExactETHForTokensSupportingFeeOnTransferTokens(
744         uint amountOutMin,
745         address[] calldata path,
746         address to,
747         uint deadline
748     )
```

SmexRouter Fork Modifications

The following **3x** modifications were found. Original PancakeSwap V2 Router contracts are represented in blue/dark blue. Modified SmexRouter contracts are represented by purple/dark purple.

The risk of unexpected functionality is Low.

```
753         ensure(deadline)
754     {
755         require(path[0] == WETH, 'PancakeRouter: INVALID_PATH');
756         uint amountIn = msg.value;
757         IWETH(WETH).deposit{value: amountIn}();
758         assert(IWETH(WETH).transfer(PancakeLibrary.pairFor(factory,
path[0], path[1]), amountIn));
759         uint balanceBefore = IERC20(path[path.length - 1]).balanceO
f(to);
760         _swapSupportingFeeOnTransferTokens(path, to);
761         require(
762             IERC20(path[path.length - 1]).balanceOf(to).sub(balance
Before) >= amountOutMin,
763             'PancakeRouter: INSUFFICIENT_OUTPUT_AMOUNT'
764         );
765     }
766     function swapExactTokensForETHSupportingFeeOnTransferTokens(
767         uint amountIn,
768         uint amountOutMin,
769         address[] calldata path,
770         address to,
771         uint deadline
772     )
773         external
```

```
753         ensure(deadline)
754     {
755         require(path[0] == WETH, 'SmexRouter: INVALID_PATH');
756         uint amountIn = msg.value;
757         IWETH(WETH).deposit{value: amountIn}();
758         assert(IWETH(WETH).transfer(SmexLibrary.pairFor(factory, pa
th[0], path[1]), amountIn));
759         uint balanceBefore = IERC20(path[path.length - 1]).balanceO
f(to);
760         _swapSupportingFeeOnTransferTokens(path, to);
761         require(
762             IERC20(path[path.length - 1]).balanceOf(to).sub(balance
Before) >= amountOutMin,
763             'SmexRouter: INSUFFICIENT_OUTPUT_AMOUNT'
764         );
765     }
766     function swapExactTokensForETHSupportingFeeOnTransferTokens(
767         uint amountIn,
768         uint amountOutMin,
769         address[] calldata path,
770         address to,
771         uint deadline
772     )
773         external
```

SmexRouter Fork Modifications

The following **4x** modifications were found. Original PancakeSwap V2 Router contracts are represented in blue/dark blue. Modified SmexRouter contracts are represented by purple/dark purple.

The risk of unexpected functionality is Low.

```
778         require(path[path.length - 1] == WETH, 'PancakeRouter: INVALID_PATH');
779         TransferHelper.safeTransferFrom(
780             path[0], msg.sender, PancakeLibrary.pairFor(factory, path[0], path[1]), amountIn
781         );
782         _swapSupportingFeeOnTransferTokens(path, address(this));
783         uint amountOut = IERC20(WETH).balanceOf(address(this));
784         require(amountOut >= amountOutMin, 'PancakeRouter: INSUFFICIENT_OUTPUT_AMOUNT');
785         IWETH(WETH).withdraw(amountOut);
786         TransferHelper.safeTransferETH(to, amountOut);
787     }
788
789     // **** LIBRARY FUNCTIONS ****
790     function quote(uint amountA, uint reserveA, uint reserveB) public pure virtual override returns (uint amountB) {
791         return PancakeLibrary.quote(amountA, reserveA, reserveB);
792     }
793
794     function getAmountOut(uint amountIn, uint reserveIn, uint reserveOut)
```

```
778         require(path[path.length - 1] == WETH, 'SmexRouter: INVALID_PATH');
779         TransferHelper.safeTransferFrom(
780             path[0], msg.sender, SmexLibrary.pairFor(factory, path[0], path[1]), amountIn
781         );
782         _swapSupportingFeeOnTransferTokens(path, address(this));
783         uint amountOut = IERC20(WETH).balanceOf(address(this));
784         require(amountOut >= amountOutMin, 'SmexRouter: INSUFFICIENT_OUTPUT_AMOUNT');
785         IWETH(WETH).withdraw(amountOut);
786         TransferHelper.safeTransferETH(to, amountOut);
787     }
788
789     // **** LIBRARY FUNCTIONS ****
790     function quote(uint amountA, uint reserveA, uint reserveB) public pure virtual override returns (uint amountB) {
791         return SmexLibrary.quote(amountA, reserveA, reserveB);
792     }
793
794     function getAmountOut(uint amountIn, uint reserveIn, uint reserveOut)
```

SmexRouter Fork Modifications

The following **4x** modifications were found. Original PancakeSwap V2 Router contracts are represented in blue/dark blue. Modified SmexRouter contracts are represented by purple/dark purple.

The risk of unexpected functionality is Low.

```
801         return PancakeLibrary.getAmountOut(amountIn, reserveIn, res
    erveOut);
802     }
803
804     function getAmountIn(uint amountOut, uint reserveIn, uint reser
    veOut)
805         public
806         pure
807         virtual
808         override
809         returns (uint amountIn)
810     {
811         return PancakeLibrary.getAmountIn(amountOut, reserveIn, res
    erveOut);
812     }
813
814     function getAmountsOut(uint amountIn, address[] memory path)
815         public
816         view
817         virtual
818         override
819         returns (uint[] memory amounts)
820     {
821         return PancakeLibrary.getAmountsOut(factory, amountIn, pat
    h);
822     }
823
824     function getAmountsIn(uint amountOut, address[] memory path)
825         public
826         view
827         virtual
828         override
829         returns (uint[] memory amounts)
830     {
831         return PancakeLibrary.getAmountsIn(factory, amountOut, pat
    h);
832     }
```

```
801         return SmexLibrary.getAmountOut(amountIn, reserveIn, reserv
    eOut);
802     }
803
804     function getAmountIn(uint amountOut, uint reserveIn, uint reser
    veOut)
805         public
806         pure
807         virtual
808         override
809         returns (uint amountIn)
810     {
811         return SmexLibrary.getAmountIn(amountOut, reserveIn, reserv
    eOut);
812     }
813
814     function getAmountsOut(uint amountIn, address[] memory path)
815         public
816         view
817         virtual
818         override
819         returns (uint[] memory amounts)
820     {
821         return SmexLibrary.getAmountsOut(factory, amountIn, path);
822     }
823
824     function getAmountsIn(uint amountOut, address[] memory path)
825         public
826         view
827         virtual
828         override
829         returns (uint[] memory amounts)
830     {
831         return SmexLibrary.getAmountsIn(factory, amountOut, path);
832     }
```

Final Thoughts

Dessert Finance has identified that the updated SmexRouter contracts have been forked from the correct PancakeSwap V2 Router contracts. All original files have been sourced and validated.

Overall, the risk rating for the fork is **Low** as the only changes appear to be naming conventions with proper follow-through in the entire contract.

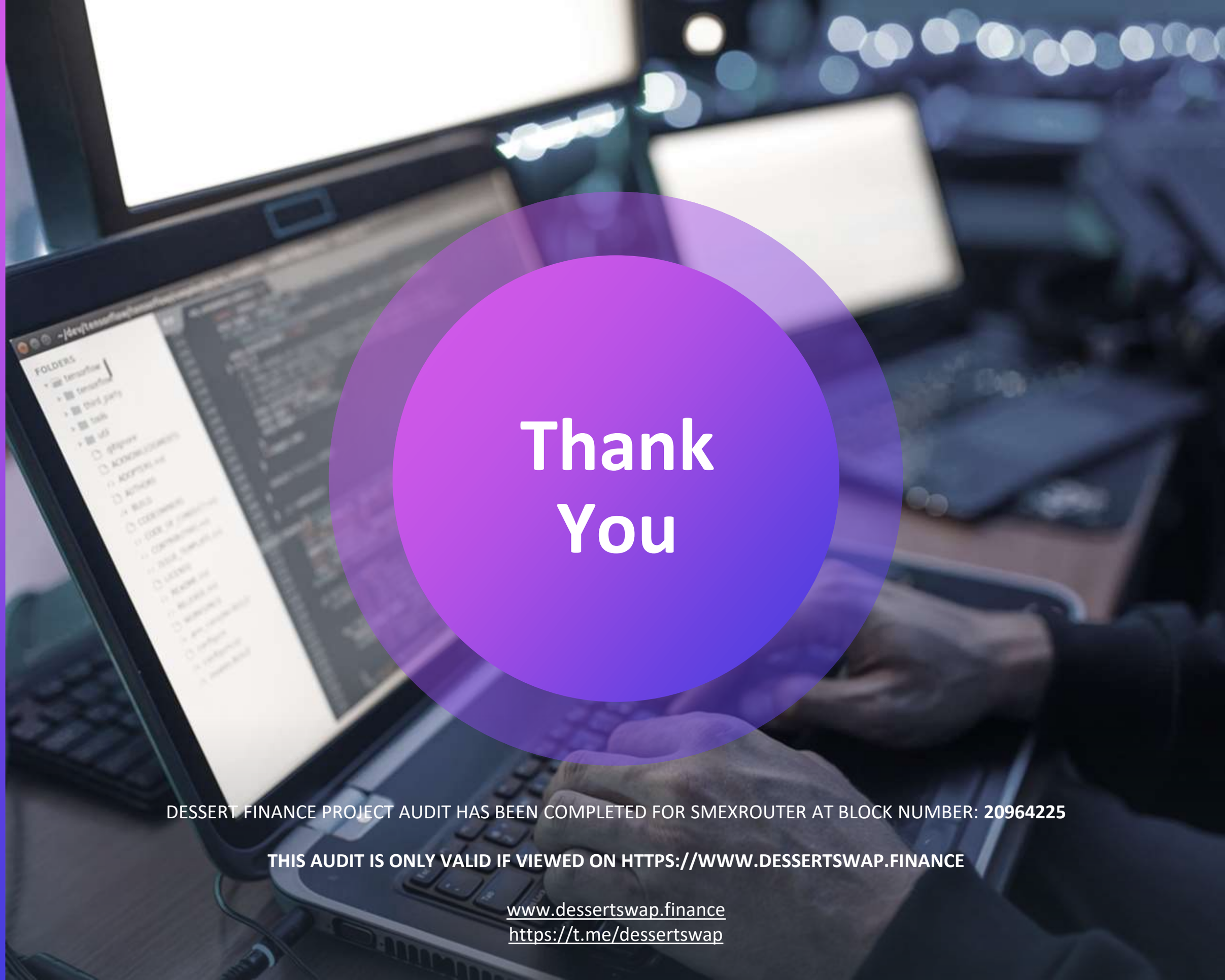
Disclaimer

The opinions expressed in this document are for general informational purposes only and are **not intended to provide specific advice or recommendations for any individual or on any specific investment**. It is only intended to provide education and public knowledge regarding projects. This audit is only applied to the type of auditing specified in this report and the scope of given in the results. Other unknown security vulnerabilities are beyond responsibility. Dessert Finance only issues this report based on the attacks or vulnerabilities that already existed or occurred before the issuance of this report. For the emergence of new attacks or vulnerabilities that exist or occur in the future, Dessert Finance lacks the capability to judge its possible impact on the security status of smart contracts, thus taking no responsibility for them. The smart contract analysis and other contents of this report are based solely on the documents and materials that the contract provider has provided to Dessert Finance or was publicly available before the issuance of this report (issuance of report recorded via block number on cover page), if the documents and materials provided by the contract provider are missing, tampered, deleted, concealed or reflected in a situation that is inconsistent with the actual situation, or if the documents and materials provided are changed after the issuance of this report, Dessert Finance assumes no responsibility for the resulting loss or adverse effects. Due to the technical limitations of any organization, this report conducted by Dessert Finance still has the possibility that the entire risk cannot be completely detected. Dessert Finance disclaims any liability for the resulting losses.

Dessert Finance provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Even projects with a low risk score have been known to pull liquidity, sell all team tokens, or exit-scam. Please exercise caution when dealing with any cryptocurrency related platforms.

The final interpretation of this statement belongs to Dessert Finance.

Dessert Finance highly advises against using cryptocurrencies as speculative investments and they should be used solely for the utility they aim to provide.



Thank You

DESSERT FINANCE PROJECT AUDIT HAS BEEN COMPLETED FOR SMEXROUTER AT BLOCK NUMBER: **20964225**

THIS AUDIT IS ONLY VALID IF VIEWED ON HTTPS://WWW.DESSERTSWAP.FINANCE

www.dessertswap.finance
https://t.me/dessertswap