

SmexRouter.sol V2

Fork Validation Report

Completed at block 20964225

PERFORMED BY DESSERT FINANCE FOR CONTRACT ADDRESS: PROVIDED SOL FILES

1

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



- 1. Contract Code Audit Overview
- 2. Fork Validation Report
- 3. Final Thoughts
- 4. Disclaimers

3

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

5

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.

<pre>35 // File: contracts\interfaces\IPancakeRouter01.sol</pre>	35 // File: contracts\interfaces\ISmexRouter01.sol
36	36
37 pragma solidity >=0.6.2;	<pre>37 pragma solidity >=0.6.2;</pre>
38	38
39 interface IPancakeRouter01 {	39 interface ISmexRouter01 {
40 function factory() external pure returns (address);	40 function factory() external pure returns (address);
41 function WETH() external pure returns (address);	41 function WETH() external pure returns (address);
42	42
43 function addLiquidity(43 function addLiquidity(
44 address tokenA,	44 address tokenA,
45 address tokenB,	45 address tokenB,
46 uint amountADesired,	46 uint amountADesired,
47 uint amountBDesired,	47 uint amountBDesired,
48 uint amountAMin,	48 uint amountAMin,
49 uint amountBMin,	49 uint amountBMin,
50 address to,	50 address to,
51 uint deadline	51 uint deadline
52) external returns (uint amountA, uint amountB, uint liquidit	52) external returns (uint amountA, uint amountB, uint liquidit
y);	y);

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.

L33	<pre>// File: contracts\interfaces\IPancakeRouter02.sol</pre>	133	<pre>// File: contracts\interfaces\ISmexRouter02.sol</pre>
L34		134	
L35	pragma solidity >=0.6.2;	135	pragma solidity >=0.6.2;
L36		136	
L37	<pre>interface IPancakeRouter02 is IPancakeRouter01 {</pre>	137	interface ISmexRouter02 is ISmexRouter01 {
L38	<pre>function removeLiquidityETHSupportingFeeOnTransferTokens(</pre>	138	<pre>function removeLiquidityETHSupportingFeeOnTransferTokens(</pre>
L39	address token,	139	address token,
L40	uint liquidity,	140	uint liquidity,
141	uint amountTokenMin,	141	uint amountTokenMin,
L42	uint amountETHMin,	142	uint amountETHMin,
L43	address to,	143	address to,
L44	uint deadline	144	uint deadline
L45) external returns (uint amountETH);	145) external returns (uint amountETH);
L46	function removeLiquidityETHWithPermitSupportingFeeOnTransferTok	146	function removeLiquidityETHWithPermitSupportingFeeOnTransferTok
	ens(ens(
147	address token,	147	address token,
L48	uint liquidity,	148	uint liquidity,
L49	uint amountTokenMin,	149	uint amountTokenMin,
150	uint amountETHMin	150	uint amountETHMin



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.

<pre>78 // File: contracts\interfaces\IPancakeFactory.sol</pre>	178 // File: contracts\interfaces\ISmexFactory.sol
79	179
80 pragma solidity >=0.5.0;	180 pragma solidity >=0.5.0;
31	181
82 interface IPancakeFactory {	182 interface ISmexFactory {
83 event PairCreated(address indexed token0, address indexed token	183 event PairCreated(address indexed token0, address indexed token
1, address pair, uint);	1, address pair, uint);
84	184
85 function feeTo() external view returns (address);	185 function feeTo() external view returns (address);
86 function feeToSetter() external view returns (address);	186 function feeToSetter() external view returns (address);
87	187
58 function getPair(address tokenA, address tokenB) external view	188 function getPair(address tokenA, address tokenB) external view
returns (address pair);	returns (address pair);
89 function allPairs(uint) external view returns (address pair);	189 function allPairs(uint) external view returns (address pair);
90 function allPairsLength() external view returns (uint);	190 function allPairsLength() external view returns (uint);
91	191
92 function createPair(address tokenA, address tokenB) external re	192 function createPair(address tokenA, address tokenB) external re
turns (address pair);	turns (address pair);
93	193
94 function setFeeTo(address) external:	<pre>194 function setFeeTo(address) external:</pre>

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.

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

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.

	E
<pre>275 // File: contracts\libraries\PancakeLibrary.sol</pre>	275 // File: contracts\libraries\SmexLibrary.sol
276	276
277 pragma solidity >=0.5.0;	277 pragma solidity >=0.5.0;
278	278
279	279
280	280
281 library PancakeLibrary {	281 library SmexLibrary {
<pre>282 using SafeMath for uint;</pre>	282 using SafeMath for uint;
283	283
284 // returns sorted token addresses, used to handle return values	284 // returns sorted token addresses, used to handle return value
from pairs sorted in this order	from pairs sorted in this order
function sortTokens(address tokenA, address tokenB) internal pu	285 function sortTokens(address tokenA, address tokenB) internal p
re returns (address token0, address token1) {	re returns (address token0, address token1) {
<pre>86 require(tokenA != tokenB, 'PancakeLibrary: IDENTICAL_ADDRES 65551);</pre>	<pre>286 require(tokenA != tokenB, 'SmexLibrary: IDENTICAL_ADDRESSE</pre>
<pre>SES'); 87 (token0, token1) = tokenA < tokenB ? (tokenA, tokenB) : (to</pre>	<pre>5'); 287 (token0, token1) = tokenA < tokenB ? (tokenA, tokenB) : (t</pre>
<pre>kenB, tokenA);</pre>	kenB, tokenA);
require(token0 != address(0), 'PancakeLibrary: ZERO_ADDRES	288 require(token0 != address(0), 'SmexLibrary: ZERO ADDRESS')
S');	
89 }	289 }
90	290
91 // calculates the CREATE2 address for a pair without making any	291 // calculates the CREATE2 address for a pair without making an
external calls	external calls
92 function pairFor(address factory, address tokenA, address token	292 function pairFor(address factory, address tokenA, address toke
B) internal pure returns (address pair) {	B) internal pure returns (address pair) {
93 (address token0, address token1) = sortTokens(tokenA, token	293 (address token0, address token1) = sortTokens(tokenA, toke
B);	B);
<pre>94 pair = address(uint(keccak256(abi.encodePacked(</pre>	<pre>294 pair = address(uint(keccak256(abi.encodePacked(</pre>
95 hex'ff',	295 hex'ff',
96 factory,	296 factory,
<pre>97 keccak256(abi.encodePacked(token0, token1)),</pre>	<pre>297 keccak256(abi.encodePacked(token0, token1)),</pre>
98 hex'00fb7f630766e6a796048ea87d01acd3068e8ff67d07814	298 hex'0406938e4eb92afe1cb10d5fd797c56b2d40da07ad6a1a
8a3fa3f4a84f69bd5' // init code hash	b826c63ea1e350abd' // init code hash
99))));	299))));
00 }	300 }
	301
02 // fetches and sorts the reserves for a pair	302 // fetches and sorts the reserves for a pair
03 function getReserves(address factory, address tokenA, address tokenA) interest (with account of the second address) (303 function getReserves(address factory, address tokenA, address
okenB) internal view returns (uint reserveA, uint reserveB) {	okenB) internal view returns (uint reserveA, uint reserveB) {
<pre>304 (address token0,) = sortTokens(tokenA, tokenB); 305 pairFor(factory, tokenA, tokenB);</pre>	<pre>304 (address token0,) = sortTokens(tokenA, tokenB); 305 pairFor(factory, tokenA, tokenB);</pre>
	<pre>305 pairFor(factory, tokenA, tokenB); 306 (uint reserve0, uint reserve1,) = ISmexPair(pairFor(factor)</pre>
<pre>006 (uint reserve0, uint reserve1,) = IPancakePair(pairFor(fact ory, tokenA, tokenB)).getReserves();</pre>	<pre>y, tokenA, tokenB)).getReserves();</pre>
(reserveA, reserveB) = tokenA == token0 ? (reserve0, reserv	<pre>307 (reserveA, reserveB) = tokenA == token0 ? (reserve0, reserveB)</pre>
e1) : (reserve1, reserve0);	e1) : (reserve1, reserve0);
(, , , , , , , , , , , , , , , , , , ,	ci, i (rescriver) rescrived);

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.

	rmai pure recurns (uinc amounco) (rnai pure recurns (uinc amounco) (
312	require(amountA > 0, 'PancakeLibrary: INSUFFICIENT_AMOUN	312	<pre>require(amountA > 0, 'SmexLibrary: INSUFFICIENT_AMOUNT');</pre>
	T');		
313	require(reserveA > 0 && reserveB > 0,	313	<pre>require(reserveA > 0 && reserveB > 0, 'SmexLibrary: INSUFFI</pre>
	FFICIENT_LIQUIDITY');		CIENT_LIQUIDITY');
314	<pre>amountB = amountA.mul(reserveB) / reserveA;</pre>	314	<pre>amountB = amountA.mul(reserveB) / reserveA;</pre>
315	}	315	}
316		316	
317	<pre>// given an input amount of an asset and pair reserves, returns</pre>	317	// given an input amount of an asset and pair reserves, returns
	the maximum output amount of the other asset		the maximum output amount of the other asset
318	function getAmountOut(uint amountIn, uint reserveIn, uint reser	318	function getAmountOut(uint amountIn, uint reserveIn, uint reser
	veOut) internal pure returns (uint amountOut) {		veOut) internal pure returns (uint amountOut) {
319	require(amountIn > 0, 'PancakeLibrary: INSUFFICIENT_INPUT_A	319	require(amountIn > 0, 'SmexLibrary: INSUFFICIENT_INPUT_AMOU
	MOUNT');		NT');
320	require(reserveIn > 0 && reserveOut > 0, 'PancakeLibrary: I	320	require(reserveIn > 0 && reserveOut > 0, 'SmexLibrary: INSU
	NSUFFICIENT_LIQUIDITY');		FFICIENT_LIQUIDITY');
321	uint amountInWithFee = amountIn.mul(9975);	321	<pre>uint amountInWithFee = amountIn.mul(9975);</pre>
322	<pre>uint numerator = amountInWithFee.mul(reserveOut);</pre>	322	<pre>uint numerator = amountInWithFee.mul(reserveOut);</pre>
323	uint denominator = reserveIn.mul(10000).add(amountInWithFe	323	uint denominator = reserveIn.mul(10000).add(amountInWithFe
	e);		e);
324	amountOut = numerator / denominator;	324	amountOut = numerator / denominator;
325	}	325	}
326		326	
327	// given an output amount of an asset and pair reserves, return	327	<pre>// given an output amount of an asset and pair reserves, return</pre>
	s a required input amount of the other asset		s a required input amount of the other asset
328	function getAmountIn(uint amountOut, uint reserveIn, uint reser	328	function getAmountIn(uint amountOut, uint reserveIn, uint reser
	<pre>veOut) internal pure returns (uint amountIn) {</pre>		veOut) internal pure returns (uint amountIn) {
329	require(amountOut > 0,	329	<pre>require(amountOut > 0, 'SmexLibrary: INSUFFICIENT_OUTPUT_AM</pre>
	_AMOUNT');		OUNT');
330	require(reserveIn > 0 && reserveOut > 0,	330	<pre>require(reserveIn > 0 && reserveOut > 0, 'SmexLibrary: INSU</pre>
	NSUFFICIENT_LIQUIDITY');		FFICIENT_LIQUIDITY');
331	<pre>uint numerator = reserveIn.mul(amountOut).mul(10000);</pre>	331	<pre>uint numerator = reserveIn.mul(amountOut).mul(10000);</pre>
332	<pre>uint denominator = reserveOut.sub(amountOut).mul(9975);</pre>	332	<pre>uint denominator = reserveOut.sub(amountOut).mul(9975);</pre>
333	amountIn = (numerator / denominator).add(1);	333	<pre>amountIn = (numerator / denominator).add(1);</pre>
	•		•

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.

	[] memory party incrimer view records (dance] memory dimonstray ([] memory poend incernar rices records (arrell memory amounted) (
338	require(path.length >= 2, <mark>'PancakeLibrary:</mark> INVALID_PATH');	338	<pre>require(path.length >= 2, 'SmexLibrary: INVALID_PATH');</pre>
339	amounts = new uint[](path.length);	339	amounts = new uint[](path.length);
340	amounts[0] = amountIn;	340	amounts[0] = amountIn;
341	for (uint i; i < path.length - 1; i++) {	341	for (uint i; i < path.length - 1; i++) {
342	(uint reserveIn, uint reserveOut) = getReserves(factor	342	(uint reserveIn, uint reserveOut) = getReserves(factor
	y, path[i], path[i + 1]);		y, path[i], path[i + 1]);
343	amounts[i + 1] = getAmountOut(amounts[i], reserveIn, re	343	amounts[i + 1] = getAmountOut(amounts[i], reserveIn, re
	serveOut);		serveOut);
344	}	344	}
345	}	345	}
346		346	
347	<pre>// performs chained getAmountIn calculations on any number of p</pre>	347	<pre>// performs chained getAmountIn calculations on any number of p</pre>
	airs		airs
348	function getAmountsIn(address factory, uint amountOut, address	348	function getAmountsIn(address factory, uint amountOut, address
	<pre>[] memory path) internal view returns (uint[] memory amounts) {</pre>		<pre>[] memory path) internal view returns (uint[] memory amounts) {</pre>
349	require(path.length >= 2, <mark>'PancakeLibrary:</mark> INVALID_PATH');	349	<pre>require(path.length >= 2, 'SmexLibrary: INVALID_PATH');</pre>
350	amounts = new uint[](path.length);	350	amounts = new uint[](path.length);
351	amounts[amounts.length - 1] = amountOut;	351	amounts[amounts.length - 1] = amountOut;
352	for (uint i = path.length - 1; i > 0; i) {	352	for (uint i = path.length - 1; $i > 0$; i) {
353	(uint reserveIn, uint reserveOut) = getReserves(factor	353	(uint reserveIn, uint reserveOut) = getReserves(factor
	<pre>y, path[i - 1], path[i]);</pre>		<pre>y, path[i - 1], path[i]);</pre>
35/	amounts[i = 1] = getAmountIn/amounts[i] _ necenveIn _ nec	354	amounts[i = 1] = getAmountIn(amounts[i] _ recerveIn _ rec

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.

388	388
389 // File: contracts\PancakeRouter.sol	389 // File: contracts\SmexRouter.sol
390	390
391 pragma solidity =0.6.6;	391 pragma solidity =0.6.6;
392	392
393	393
394	394
395	395
396	396
397	397
398	398
399 contract PancakeRouter is IPancakeRouter02 {	399 contract SmexRouter is ISmexRouter02 {
400 using SafeMath for uint;	400 using SafeMath for uint;
401	401
402 address public immutable override factory;	402 address public immutable override factory;
<pre>403 address public immutable override WETH;</pre>	403 address public immutable override WETH;
404 405 modifier ensure(uint deadline) {	404 405 modifier ensure(uint deadline) {
405 require(deadline >= block.timestamp, 'PancakeRouter: EXPIRE	405 require(deadline >= block.timestamp, 'SmexRouter: EXPIRE
D');	D');
407 _;	407 _;
408 }	408 }
409	409
410 constructor(address _factory, address _WETH) public {	<pre>410 constructor(address _factory, address _WETH) public {</pre>
411 factory = _factory;	411 factory = _factory;
412 WETH = _WETH;	412 WETH = _WETH;
413 }	413 }
414	414

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.

428	<pre>// create the pair if it doesn't exist yet</pre>	428	<pre>// create the pair if it doesn't exist yet</pre>
429	<pre>if (IPancakeFactory(factory).getPair(tokenA, tokenB) == add</pre>	429	<pre>if (ISmexFactory(factory).getPair(tokenA, tokenB) == addres</pre>
	ress(0)) {		s(0)) {
430	<pre>IPancakeFactory(factory).createPair(tokenA, tokenB);</pre>	430	ISmexFactory(factory).createPair(tokenA, tokenB);
431	}	431	}
432	(uint reserveA, uint reserveB) = PancakeLibrary.getReserves	432	<pre>(uint reserveA, uint reserveB) = SmexLibrary.getReserves(fa</pre>
	(factory, tokenA, tokenB);		ctory, tokenA, tokenB);
433	if (reserveA == 0 && reserveB == 0) {	433	if (reserveA == 0 && reserveB == 0) {
434	(amountA, amountB) = (amountADesired, amountBDesired);	434	(amountA, amountB) = (amountADesired, amountBDesired);
435	} else {	435	} else {
436	uint amountBOptimal = PancakeLibrary.quote(amountADesir	436	<pre>uint amountBOptimal = SmexLibrary.quote(amountADesired,</pre>
e	ed, reserveA, reserveB);		reserveA, reserveB);
437	if (amountBOptimal <= amountBDesired) {	437	if (amountBOptimal <= amountBDesired) {
438	require(amountBOptimal >= amountBMin, <mark>'PancakeRoute</mark>	438	<pre>require(amountBOptimal >= amountBMin, 'SmexRouter:</pre>
	<pre>r: INSUFFICIENT_B_AMOUNT');</pre>		INSUFFICIENT_B_AMOUNT');
439	(amountA, amountB) = (amountADesired, amountBOptima	439	(amountA, amountB) = (amountADesired, amountBOptima
	1);		1);
440	} else {	440	} else {
441	<pre>uint amountAOptimal = PancakeLibrary.quote(amountBD</pre>	441	<pre>uint amountAOptimal = SmexLibrary.quote(amountBDesi</pre>
	esired, reserveB, reserveA);		red, reserveB, reserveA);
442	assert(amountAOptimal <= amountADesired);	442	<pre>assert(amountAOptimal <= amountADesired);</pre>
443	require(amountAOptimal >= amountAMin, 'PancakeRoute	443	require(amountAOptimal >= amountAMin, 'SmexRouter:
	<pre>INSUFFICIENT_A_AMOUNT');</pre>		INSUFFICIENT_A_AMOUNT');
444	(amountA, amountB) = (amountAOptimal, amountBDesire	444	(amountA, amountB) = (amountAOptimal, amountBDesire
445	d);	445	d);
	1		1
446	i .	446	1
447		447	
448	function addLiquidity(448	function addLiquidity(
	··· · ·		

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.

458	(amounta, amountb) = _addLlquldlty(tokena, tokenb, amountab esired, amountBDesired, amountAMin, amountBMin);	458	<pre>(amountA, amountB) = _addLlquldlty(tokenA, tokenB, amountAD esired, amountBDesired, amountAMin, amountBMin);</pre>
459	address pair = PancakeLibrary.pairFor(factory, tokenA, toke	459	address pair = SmexLibrary.pairFor(factory, tokenA, token
439	nB);	439	B);
460	TransferHelper.safeTransferFrom(tokenA, msg.sender, pair, a	460	TransferHelper.safeTransferFrom(tokenA, msg.sender, pair, a
400	mountA);	400	mountA);
461	TransferHelper.safeTransferFrom(tokenB, msg.sender, pair, a	461	TransferHelper.safeTransferFrom(tokenB, msg.sender, pair, a
101	mountB);	101	mountB);
462	liquidity = IPancakePair(pair).mint(to);	462	liquidity = ISmexPair(pair).mint(to);
463	}	463	}
464	function addLiquidityETH(464	function addLiquidityETH(
465	address token,	465	address token,
466	uint amountTokenDesired,	466	uint amountTokenDesired,
467	uint amountTokenMin,	467	uint amountTokenMin,
468	uint amountETHMin,	468	uint amountETHMin,
469	address to,	469	address to,
470	uint deadline	470	uint deadline
471) external virtual override payable ensure(deadline) returns (u	471) external virtual override payable ensure(deadline) returns (u
	int amountToken, uint amountETH, uint liquidity) {		int amountToken, uint amountETH, uint liquidity) {
472	(amountToken, amountETH) = _addLiquidity(472	(amountToken, amountETH) = _addLiquidity(
473	token,	473	token,
474	WETH,	474	WETH,
475	amountTokenDesired,	475	amountTokenDesired,
476	msg.value,	476	msg.value,
477	amountTokenMin,	477	amountTokenMin,
478	amountETHMin	478	amountETHMin
479);	479);
480	address pair = PancakeLibrary.pairFor(factory, token, WET	480	address pair = SmexLibrary.pairFor(factory, token, WETH);
	H);		
481	TransferHelper.safeTransferFrom(token, msg.sender, pair, am	481	TransferHelper.safeTransferFrom(token, msg.sender, pair, am
	ountToken);		ountToken);
482	IWETH(WETH).deposit{value: amountETH}();	482	IWETH(WETH).deposit{value: amountETH}();
483	<pre>assert(IWETH(WETH).transfer(pair, amountETH));</pre>	483	<pre>assert(IWETH(WETH).transfer(pair, amountETH));</pre>
484	<pre>liquidity = IPancakePair(pair).mint(to);</pre>	484	<pre>liquidity = ISmexPair(pair).mint(to);</pre>
485	// refund dust eth, if any	485	// refund dust eth, if any
486	if (msg.value > amountETH) TransferHelper.safeTransferETH(m	486	if (msg.value > amountETH) TransferHelper.safeTransferETH(m
	sg.sender, msg.value - amountETH);		sg.sender, msg.value - amountETH);
	×		•

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.

	A, uint amountB) {	1	A, uint amountB) {
499	<pre>address pair = PancakeLibrary.pairFor(factory, tokenA, toke</pre>	499	address pair = SmexLibrary.pairFor(factory, tokenA, token
	nB);		B);
500	IPancakePair(pair).transferFrom(msg.sender, pair, liquidit	500	ISmexPair(pair).transferFrom(msg.sender, pair, liquidity);
	y); // send liquidity to pair		// send liquidity to pair
501	<pre>(uint amount0, uint amount1) = IPancakePair(pair).burn(to);</pre>	501	<pre>(uint amount0, uint amount1) = ISmexPair(pair).burn(to);</pre>
502	<pre>(address token0,) = PancakeLibrary.sortTokens(tokenA, token</pre>	502	<pre>(address token0,) = SmexLibrary.sortTokens(tokenA, tokenB);</pre>
	B);		
503	(amountA, amountB) = tokenA == token0 ? (amount0, amount1)	503	(amountA, amountB) = tokenA == token0 ? (amountO, amount1)
	: (amount1, amount0);		: (amount1, amount0);
504	require(amountA >= amountAMin, <mark>'PancakeRouter:</mark> INSUFFICIENT	504	<pre>require(amountA >= amountAMin, 'SmexRouter: INSUFFICIENT_A_</pre>
	_A_AMOUNT');	1	AMOUNT');
505	require(amountB >= amountBMin,	505	<pre>require(amountB >= amountBMin, 'SmexRouter: INSUFFICIENT_B_</pre>
	_B_AMOUNT');		AMOUNT');
506	}	506	}
507	function removeLiquidityETH(507	function removeLiquidityETH(
508	address token,	508	address token,
509	uint liquidity,	509	uint liquidity,
510	uint amountTokenMin,	510	uint amountTokenMin,
511	uint amountETHMin,	511	uint amountETHMin,
512	address to,	512	address to,

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.

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

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.

588) external virtual override returns (uint amountETH) {	588) external virtual override returns (uint amountETH) {
589 address pair = PancakeLibrary.pairFor(factory, token, WET	T 589 address pair = SmexLibrary.pairFor(factory, token, WETH);
H);	
uint value = approveMax ? uint(-1) : liquidity;	590 uint value = approveMax ? uint(-1) : liquidity;
591 IPancakePair(pair).permit(msg.sender, address(this), valu	ue, 591 ISmexPair(pair).permit(msg.sender, address(this), value, de
deadline, v, r, s);	adline, v, r, s);
i92 amountETH = removeLiquidityETHSupportingFeeOnTransferToke	ens 592 amountETH = removeLiquidityETHSupportingFeeOnTransferTokens
((
93 token, liquidity, amountTokenMin, amountETHMin, to, d	
dline	dline
94);	594);
95 }	595 }
96	596
97 // **** SWAP ****	597 // **** SWAP ****
98 // requires the initial amount to have already been sent to t	
first pair	first pair
99 function _swap(uint[] memory amounts, address[] memory path,	
dress _to) internal virtual {	dress _to) internal virtual {
00 for (uint i; i < path.length - 1; i++) {	600 for (uint i; i < path.length - 1; i++) {
01 (address input, address output) = (path[i], path[i +	
1]);	1]);
<pre>02 (address token0,) = PancakeLibrary.sortTokens(input,</pre>	
tput);	t);
<pre>03 uint amountOut = amounts[i + 1]; (vist amountOout vist amount10:t) issue taleno</pre>	603 uint amountOut = amounts[i + 1];
<pre>04 (uint amount00ut, uint amount10ut) = input == token0 (uint(0), amount0ut) : (amount0ut, uint(0));</pre>	<pre>0 ? 604 (uint amount00ut, uint amount10ut) = input == token0 ? (uint(0), amount0ut) : (amount0ut, uint(0));</pre>
address to = i < path.length - 2 ? PancakeLibrary.pai	
or(factory, output, path[i + 2]) : _to;	(factory, output, path[i + 2]) : _to;
06 IPancakePair(PancakeLibrary.pairFor(factory, input, or pairFor(factory))	
put)).swap(swap(
07 amount00ut, amount10ut, to, new bytes(0)	607 amount00ut, amount10ut, to, new bytes(0)
508);	608);
///	/)

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.

61/) external virtual override ensure(deadline) returns (uint[] me	61/
	mory amounts) {	
618	amounts = PancakeLibrary.getAmountsOut(factory, amountIn, p	618
	ath);	
619	<pre>require(amounts[amounts.length - 1] >= amountOutMin, 'Panca</pre>	619
	<pre>keRouter: INSUFFICIENT_OUTPUT_AMOUNT');</pre>	
620	TransferHelper.safeTransferFrom(620
621	<pre>path[0], msg.sender, PancakeLibrary.pairFor(factory, pa</pre>	621
	th[0], path[1]), amounts[0]	
622);	622
623	_swap(amounts, path, to);	623
624	}	624
625	function swapTokensForExactTokens(625
626	uint amountOut,	626
627	uint amountInMax,	627
628	address[] calldata path,	628
629	address to,	629
630	uint deadline	630
631) external virtual override ensure(deadline) returns (uint[] me	631
	mory amounts) {	
632	amounts = PancakeLibrary.getAmountsIn(factory, amountOut, p	632
	ath);	
633	require(amounts[0] <= amountInMax,	633
	E_INPUT_AMOUNT');	
634	TransferHelper.safeTransferFrom(634
635	path[0], msg.sender, PancakeLibrary.pairFor(factory, pa	635
	th[0], path[1]), amounts[0]	
636);	636
637	_swap(amounts, path, to);	637
638	}	638
639	function swapExactETHForTokens(uint amountOutMin, address[] cal	639

) external virtual override ensure(deadline) returns (uint[] me mory amounts) { amounts = SmexLibrary.getAmountsOut(factory, amountIn, pat h); require(amounts[amounts.length - 1] >= amountOutMin, 'SmexR outer: INSUFFICIENT_OUTPUT_AMOUNT'); TransferHelper.safeTransferFrom(path[0], msg.sender, SmexLibrary.pairFor(factory, path [0], path[1]), amounts[0]); _swap(amounts, path, to); function swapTokensForExactTokens(uint amountOut, uint amountInMax, address[] calldata path, address to, uint deadline) external virtual override ensure(deadline) returns (uint[] me mory amounts) { amounts = SmexLibrary.getAmountsIn(factory, amountOut, pat h); require(amounts[0] <= amountInMax, 'SmexRouter: EXCESSIVE_I</pre> NPUT AMOUNT'); TransferHelper.safeTransferFrom(path[0], msg.sender, SmexLibrary.pairFor(factory, path [0], path[1]), amounts[0]); _swap(amounts, path, to); 9

and the set of the second set of

function swapExactETHForTokens(uint amountOutMin, address[] cal . . . and the second second second

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.

)40	recurns (utric[] memory amounts)	040 recurns (uinc[] memory amounts)
546	{	646 {
547	require(path[0] == WETH,	<pre>647 require(path[0] == WETH, 'SmexRouter: INVALID_PATH');</pre>
i48	<pre>amounts = PancakeLibrary.getAmountsOut(factory, msg.value,</pre>	648 amounts = SmexLibrary.getAmountsOut(factory, msg.value, pat
	path);	h);
549	<pre>require(amounts[amounts.length - 1] >= amountOutMin, 'Panca</pre>	649 require(amounts[amounts.length - 1] >= amountOutMin, 'SmexR
	<pre>keRouter: INSUFFICIENT_OUTPUT_AMOUNT');</pre>	outer: INSUFFICIENT_OUTPUT_AMOUNT');
50	<pre>IWETH(WETH).deposit{value: amounts[0]}();</pre>	<pre>650 IWETH(WETH).deposit{value: amounts[0]}();</pre>
i51	<pre>assert(IWETH(WETH).transfer(PancakeLibrary.pairFor(factory,</pre>	651 assert(IWETH(WETH).transfer(SmexLibrary.pairFor(factory, pa
	path[0], path[1]), amounts[0]));	th[0], path[1]), amounts[0]));
52	_swap(amounts, path, to);	<pre>652 _swap(amounts, path, to);</pre>
553	}	653 }
54	function swapTokensForExactETH(uint amountOut, uint amountInMa	654 function swapTokensForExactETH(uint amountOut, uint amountInMa
	x, address[] calldata path, address to, uint deadline)	x, address[] calldata path, address to, uint deadline)
555	external	655 external
556	virtual	656 virtual
557	override	657 override
558	ensure(deadline)	658 ensure(deadline)
559	returns (uint[] memory amounts)	659 returns (uint[] memory amounts)
60	{	660 {
i 61	require(path[path.length - 1] == WETH,	661 require(path[path.length - 1] == WETH, 'SmexRouter: INVALID
	LID_PATH');	_PATH');
62	<pre>amounts = PancakeLibrary.getAmountsIn(factory, amountOut, p</pre>	662 amounts = SmexLibrary.getAmountsIn(factory, amountOut, pat
	ath);	h);
i63	<pre>require(amounts[0] <= amountInMax, 'PancakeRouter: EXCESSIV</pre>	663 require(amounts[0] <= amountInMax, 'SmexRouter: EXCESSIVE_I
	E_INPUT_AMOUNT');	NPUT_AMOUNT');
64	TransferHelper.safeTransferFrom(664 TransferHelper.safeTransferFrom(
65	<pre>path[0], msg.sender, PancakeLibrary.pairFor(factory, pa </pre>	665 path[0], msg.sender, SmexLibrary.pairFor(factory, path
	th[0], path[1]), amounts[0]	[0], path[1]), amounts[0]
666);	666);
67	_swap(amounts, path, address(this));	<pre>667 _swap(amounts, path, address(this));</pre>

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.

	<pre>[path.length - 1] == WETH, 'PancakeRouter: INVA</pre>	678	require(path[path.length - 1] == WETH, 'SmexRouter: INVALID
LID_PATH');			_PATH');
<pre>679 amounts = Pa ath);</pre>	<pre>ncakeLibrary.getAmountsOut(factory, amountIn, p</pre>	679	<pre>amounts = SmexLibrary.getAmountsOut(factory, amountIn, pat h);</pre>
-	<pre>nts[amounts.length - 1] >= amountOutMin, 'Panca</pre>	680	
keRouter: INSUFFICIE			outer: INSUFFICIENT OUTPUT AMOUNT');
	er.safeTransferFrom(681	TransferHelper.safeTransferFrom(
682 path[0]	msg.sender, PancakeLibrary.pairFor(factory, pa	682	
th[0], path[1]), amo			[0], path[1]), amounts[0]
683);		683	
684 _swap(amount	s, path, address(this));	684	_swap(amounts, path, address(this));
685 IWETH(WETH).	withdraw(amounts[amounts.length - 1]);	685	<pre>IWETH(WETH).withdraw(amounts[amounts.length - 1]);</pre>
686 TransferHelp	er.safeTransferETH(to, amounts[amounts.length -	686	TransferHelper.safeTransferETH(to, amounts[amounts.length -
1]);			1]);
687 }		687	}
688 function swapETH	<pre>IForExactTokens(uint amountOut, address[] callda</pre>	688	function swapETHForExactTokens(uint amountOut, address[] callda
ta path, address to,	uint deadline)		ta path, address to, uint deadline)
689 external		689	external
690 virtual		690	virtual
691 override		691	override
692 payable		692	payable
693 ensure(dead]	ine)	693	ensure(deadline)
694 returns (uin	t[] memory amounts)	694	returns (uint[] memory amounts)
695 {		695	· · · · · · · · · · · · · · · · · · ·
	<pre>[0] == WETH, 'PancakeRouter: INVALID_PATH');</pre>	696	
697 amounts = Pa	<pre>ncakeLibrary.getAmountsIn(factory, amountOut, p</pre>	697	amounts = SmexLibrary.getAmountsIn(factory, amountOut, pat
ath);			h);
	<pre>ints[0] <= msg.value, 'PancakeRouter: EXCESSIVE_</pre>	698	
INPUT_AMOUNT');			UT_AMOUNT');
	<pre>deposit{value: amounts[0]}();</pre>	699	
	<pre>I(WETH).transfer(PancakeLibrary.pairFor(factory,</pre>	700	
path[0], path[1]), a		705	<pre>th[0], path[1]), amounts[0]));</pre>
	s, path, to);	701	
702 // refund du	ist eth if anv	702	// refund dust eth if any

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.

110	(audress input, audress output) - (path[i], path[i +	110	(audress input, audress output) - (path[i], path[i +
	1]);		1]);
711		711	
	tput);		t);
712		712	
	<pre>(factory, input, output));</pre>		input, output));
713	uint amountInput;	713	uint amountInput;
714	uint amountOutput;	714	uint amountOutput;
715	{ // scope to avoid stack too deep errors	715	{ // scope to avoid stack too deep errors
716	<pre>(uint reserve0, uint reserve1,) = pair.getReserves();</pre>	716	<pre>(uint reserve0, uint reserve1,) = pair.getReserves();</pre>
717	(uint reserveInput, uint reserveOutput) = input == toke	717	(uint reserveInput, uint reserveOutput) = input == toke
	n0 ? (reserve0, reserve1) : (reserve1, reserve0);		n0 ? (reserve0, reserve1) : (reserve1, reserve0);
718	amountInput = IERC20(input).balanceOf(address(pair)).su	718	amountInput = IERC20(input).balanceOf(address(pair)).su
	b(reserveInput);		b(reserveInput);
719	<pre>amountOutput = PancakeLibrary.getAmountOut(amountInput,</pre>	719	amountOutput = SmexLibrary.getAmountOut(amountInput, re
	reserveInput, reserveOutput);		serveInput, reserveOutput);
720	}	720	}
721	<pre>(uint amount00ut, uint amount10ut) = input == token0 ?</pre>	721	(uint amount00ut, uint amount10ut) = input == token0 ?
	<pre>(uint(0), amountOutput) : (amountOutput, uint(0));</pre>		<pre>(uint(0), amountOutput) : (amountOutput, uint(0));</pre>
722	address to = i < path.length - 2 ? <mark>PancakeLibrary.pairF</mark>	722	address to = i < path.length - 2 ? SmexLibrary.pairFor
	<pre>or(factory, output, path[i + 2]) : _to;</pre>		<pre>(factory, output, path[i + 2]) : _to;</pre>
723	<pre>pair.swap(amount00ut, amount10ut, to, new bytes(0));</pre>	723	<pre>pair.swap(amount00ut, amount10ut, to, new bytes(0));</pre>
724	}	724	}
725	}	725	}

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.

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

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.

753		ensure(deadline)	/53	ensure(deadline)
754	{		754	{
755		require(path[0] == WETH, 'PancakeRouter: INVALID_PATH');	755	<pre>require(path[0] == WETH, 'SmexRouter: INVALID_PATH');</pre>
756		uint amountIn = msg.value;	756	uint amountIn = msg.value;
757		IWETH(WETH).deposit{value: amountIn}();	757	IWETH(WETH).deposit{value: amountIn}();
758		assert(IWETH(WETH).transfer(PancakeLibrary.pairFor(factory,	758	<pre>assert(IWETH(WETH).transfer(SmexLibrary.pairFor(factory, pa</pre>
	path[0]	, path[1]), amountIn));		<pre>th[0], path[1]), amountIn));</pre>
759		uint balanceBefore = IERC20(path[path.length - 1]).balanceO	759	uint balanceBefore = IERC20(path[path.length - 1]).balanceO
	f(to);			f(to);
760		_swapSupportingFeeOnTransferTokens(path, to);	760	_swapSupportingFeeOnTransferTokens(path, to);
761		require(761	
762		IERC20(path[path.length - 1]).balanceOf(to).sub(balance	762	IERC20(path[path.length - 1]).balanceOf(to).sub(balance
	Before)	≻= amountOutMin,		Before) >= amountOutMin,
763		'PancakeRouter: INSUFFICIENT_OUTPUT_AMOUNT'	763	'SmexRouter: INSUFFICIENT_OUTPUT_AMOUNT'
764);	764);
765	}		765	}
766	fun	ction swapExactTokensForETHSupportingFeeOnTransferTokens(766	function swapExactTokensForETHSupportingFeeOnTransferTokens(
767		uint amountIn,	767	uint amountIn,
768		uint amountOutMin,	768	uint amountOutMin,
769		address[] calldata path,	769	address[] calldata path,
770		address to,	770	address to,
771		uint deadline	771	uint deadline
772)		772)
773		external	773	external

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.

<pre>778 require(path[path.length - 1] == WETH, 'PancakeRouter: INVA</pre>	<pre>778 require(path[path.length - 1] == WETH, 'SmexRouter: INVALID</pre>
LID_PATH');	_PATH');
779 TransferHelper.safeTransferFrom(779 TransferHelper.safeTransferFrom(
<pre>780 path[0], msg.sender, PancakeLibrary.pairFor(factory, pa</pre>	<pre>780 path[0], msg.sender, SmexLibrary.pairFor(factory, path</pre>
th[0], path[1]), amountIn	<pre>[0], path[1]), amountIn</pre>
781);	781);
782swapSupportingFeeOnTransferTokens(path, address(this));	782 _swapSupportingFeeOnTransferTokens(path, address(this));
<pre>783 uint amountOut = IERC20(WETH).balanceOf(address(this));</pre>	<pre>783 uint amountOut = IERC20(WETH).balanceOf(address(this));</pre>
784 require(amountOut >= amountOutMin, 'PancakeRouter: INSUFFIC	784 require(amountOut >= amountOutMin, 'SmexRouter: INSUFFICIEN
IENT_OUTPUT_AMOUNT');	T_OUTPUT_AMOUNT');
<pre>785 IWETH(WETH).withdraw(amountOut);</pre>	785 IWETH(WETH).withdraw(amountOut);
786 TransferHelper.safeTransferETH(to, amountOut);	786 TransferHelper.safeTransferETH(to, amountOut);
787 }	787 }
788	788
789 // **** LIBRARY FUNCTIONS ****	789 // **** LIBRARY FUNCTIONS ****
790 function quote(uint amountA, uint reserveA, uint reserveB) publ	790 function quote(uint amountA, uint reserveA, uint reserveB) publ
ic pure virtual override returns (uint amountB) {	ic pure virtual override returns (uint amountB) {
791 return PancakeLibrary.quote(amountA, reserveA, reserveB);	791 return SmexLibrary.quote(amountA, reserveA, reserveB);
792 }	792 }
793	793
794 function getAmountOut(uint amountIn, uint reserveIn, uint reser	794 function getAmountOut(uint amountIn, uint reserveIn, uint reser
veOut)	veOut)
	111 I.1

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.

000	ı		000	ι	
301		return PancakeLibrary.getAmountOut(amountIn, reserveIn, res	801		return SmexLibrary.getAmountOut(amountIn, reserveIn, reserv
	erveOut);		eOut);	
02	}		802	}	
03			803		
04	fun	ction getAmountIn(uint amountOut, uint reserveIn, uint reser	804		nction getAmountIn(uint amountOut, uint reserveIn, uint reser
	veOut)			veOut)	
05		public	805		public
06		pure	806		pure
07		virtual	807		virtual
08		override	808		override
09		returns (uint amountIn)	809		returns (uint amountIn)
10	{		810	{	
11		return PancakeLibrary.getAmountIn(amountOut, reserveIn, res	811		return SmexLibrary.getAmountIn(amountOut, reserveIn, reserv
	erveOut);		eOut);	
12	}		812	}	
13			813		
14	fun	ction getAmountsOut(uint amountIn, address[] memory path)	814	fur	nction getAmountsOut(uint amountIn, address[] memory path)
15		public	815		public
16		view	816		view
17		virtual	817		virtual
18		override	818		override
19		returns (uint[] memory amounts)	819		returns (uint[] memory amounts)
20	{		820	{	
21		return PancakeLibrary.getAmountsOut(factory, amountIn, pat	821		<pre>return SmexLibrary.getAmountsOut(factory, amountIn, path);</pre>
	h);				
22	}		822	}	
23			823		
24	fun	ction getAmountsIn(uint amountOut, address[] memory path)	824	fur	<pre>nction getAmountsIn(uint amountOut, address[] memory path)</pre>
25		public	825		public
26		view	826		view
327		virtual	827		virtual
328		override	828		override
329		returns (uint[] memory amounts)	829		returns (uint[] memory amounts)
30	{		830	{	
24		return PancakeLibrary.getAmountsIn(factory, amountOut, pat	831		<pre>return SmexLibrary.getAmountsIn(factory, amountOut, path);</pre>
31					
10	h);				

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.

19

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

29