

SoneFi_AMM_Swap_Analysis_Secure

3

Source From Secure3:

https://github.com/WELabsPerpCodeBase/WELabs_AMM_Swap_Audit#code-repos

Purpose

The primary objective of this analysis is to assess the code similarity between the [source repo](#) and the [target repo](#).

Our engineering team has embarked on a detailed direct comparison of the code repositories. This examination aims to accurately quantify and understand the extent of similarity.

Our analysis result can be found in the [conclusion](#).

Code Repos

Source

- <https://github.com/WELabsPerpCodeBase/AMM-Swap>
 - commit hash: 34b4a88546cb53ea40f1f2d8e56790e4de7a775d
 - [router.sol](#)
 - [factory.sol](#)

Target

- <https://etherscan.io/address/0x7a250d5630b4cf539739df2c5dacb4c659f2488d#code>
 - 0x7a250d5630b4cf539739df2c5dacb4c659f2488d
- <https://etherscan.io/address/0x074b735db7a27cf0d68f0eba6769b4c751fcfc23d#code>
 - 0x074b735db7a27cf0d68f0eba6769b4c751fcfc23d

A copy of the codes has also been cloned into [source_code/](#) and [target_code/](#) for reference.

Conclusion

Based on our comparison, we conclude that the source and the target are: Nearly Identical with a new contract added [WELabAMMSwapFactory](#).

The majority of changes are:

1. Function renames
2. Variable renames
3. Fee change
4. New contract WELabAMMSwapFactory - the added code has been extract to a file in [results/WELabAMMSwapFactory.sol](#)

With proper execution, the WELabs_AMM_Swap protocol should perform in very similar ways to Uniswap V2. Prior security assessments on target repos can be a good reference. Readers of this analysis, based on your security requirements, should determine on their own whether or not to directly trust the prior or existing security assessments or audits on target repos.

Methodology for Similarity Comparison

In our approach, we primarily employed the fundamental algorithm underlying the [diff](#) utility to ascertain the degree of similarity at the string level between the source and target repos.

The [diff](#) utility, widely used in text comparison and analysis, operates on the principle of finding the longest common subsequence (LCS) between two sets of data, typically text files:

1. Longest Common Subsequence (LCS): The LCS is the heart of the diff algorithm. It finds the longest sequence of characters that appear in the same order in both files. Unlike substrings, the characters in a subsequence are not required to occupy consecutive positions. The LCS serves as a baseline for understanding the similarities between two files.
2. Differences Identification: After determining the LCS, diff analyzes the sections of the files that don't form part of this subsequence. These segments are the differences or changes. The algorithm efficiently pinpoints where the two files diverge from the LCS, marking these as either additions, deletions, or modifications.

The screenshots are from Beyond Compare for better visualization.

Tools used:

- `diff` - <https://ss64.com/osx/diff.html>
- DiffMerge: <https://sourcegear.com/difmerge/>
- Beyond Compare: <https://www.scootersoftware.com/>

Comparison Detail

Diff Compare -> factory.sol

- Diff:
 - [source_code/AMM-Swap/contracts/core/factory.sol](#)
 - [target_code/uniswap-v2/UniswapV2Pair.sol](#)
- Output: [./results/AMM-Swap/contracts/core/factory.sol.patch](#)

```
$ diff ./source_code/AMM-Swap/contracts/core/factory.sol
./target_code/uniswap-v2/UniswapV2Pair.sol >
./results/AMM-Swap/contracts/core/factory.sol.diff.patch
```

Main changes:

1. file renames
2. variable and function renames
3. add a contract WELabAMMSwapFactory with 38 newlines.

Screenshot from Beyond Compare for Reference:

- [./source_code/AMM-Swap/contracts/core/factory.sol](#)
- TIPS: The result has already made several contracts in factory.sol change their locations.

```

// File: contracts/UniswapV2ERC20.sol
pragma solidity =0.5.16;

contract UniswapV2ERC20 is IUniswapV2ERC20 {
    using SafeMath for uint;

    string public constant name = 'Uniswap V2';
    string public constant symbol = 'UNI-V2';
    uint8 public constant decimals = 18;
    uint _publicTotalSupply;
    mapping(address => uint) public balanceOf;
    mapping(address => mapping(address => uint)) public allowance;

    emit Burn(msg.sender, amount0, amount1, to);
}

// this low-level function should be called from a contract which performs import
function swap(uint amount0Out, uint amount1Out, address to, bytes calldata data)
    require(amount0Out > 0 || amount1Out > 0, 'UniswapV2: INSUFFICIENT_OUTPUT_AMOUNT');
    (uint128 _reserve0, uint128 _reserve1,) = getReserves(); // gas savings
    require(amount0Out < _reserve0 && amount1Out < _reserve1, 'UniswapV2: INSUFFICIENT_INPUT_AMOUNT');

    uint balance0;
    uint balance1;
    { // scope for _token0, avoids stack too deep errors
        address _token0 = token0;
        address _token1 = token1;
        require(to != _token0 && to != _token1, 'UniswapV2: INVALID_TO');
        if (amount0Out > 0) _safeTransfer(_token0, to, amount0Out); // optimisticall
        if (amount1Out > 0) _safeTransfer(_token1, to, amount1Out); // optimisticall
        if (data.length > 0) IUniswapV2Callee(to).uniSwapV2Call(msg.sender, amount0Out, _token0, _token1, to, data);
        balance0 = IERC20(_token0).balanceOf(address(this));
        balance1 = IERC20(_token1).balanceOf(address(this));
    }
    uint amount0In = balance0 - _reserve0 - amount0Out ? balance0 - (_reserve0 - amount0In) : balance1 - (_reserve1 - amount1In) > 0, 'UniswapV2: INSUFFICIENT_INPUT_AMOUNT';

    { // scope for reserve{0,1}Adjusted, avoids stack too deep errors
        uint balance0Adjusted = balance0.mul(1000).sub(amount0In.mul(3));
        uint balance1Adjusted = balance1.mul(1000).sub(amount1In.mul(3));
        require(balance0Adjusted.mul(balance1Adjusted) >= uint(_reserve0).mul(_reserve1));
    }

    _update(balance0, balance1, _reserve0, _reserve1);
    emit SwapFeePaid(token0, amount0In, amount1In, amount0Out, amount1Out, to);
}

// Force balances to match reserves
function sync(address to) external lock {
    address token0 = token0;
    address token1 = token1;
    require(token0 != token1, 'UniswapV2: same tokens');
    _safeTransfer(_token0, to, IERC20(_token0).balanceOf(address(this)).sub(reserve0));
    _safeTransfer(_token1, to, IERC20(_token1).balanceOf(address(this)).sub(reserve1));
}

// Force reserves to match balances
function sync() external lock {
    IERC20(token0).balanceOf(address(this)), IERC20(token1).balanceOf(address(this)), reserve0, reserve1;
}

// Force balances to match reserves
function sync(address to) external lock {
    address token0 = token0;
    address token1 = token1;
    require(token0 != token1, 'UniswapV2: same tokens');
    _safeTransfer(_token0, to, IERC20(_token0).balanceOf(address(this)).sub(reserve0));
    _safeTransfer(_token1, to, IERC20(_token1).balanceOf(address(this)).sub(reserve1));
}

// Force reserves to match balances
function sync() external lock {
    IERC20(token0).balanceOf(address(this)), IERC20(token1).balanceOf(address(this)), reserve0, reserve1;
}

contract WELabAMMSwapERC20 is INELabAMMSwapERC20 {
    using SafeMath for uint;

    string public constant name = 'WELabAMMSwap-Liquidity Token';
    string public constant symbol = 'WELabAMMSwap-LP';
    uint8 public constant decimals = 18;
    uint _publicTotalSupply;
    mapping(address => uint) public balanceOf;
    mapping(address => mapping(address => uint)) public allowance;

    emit Burn(msg.sender, amount0, amount1, to);
}

// this low-level function should be called from a contract which performs import
function swap(uint amount0Out, uint amount1Out, address to, bytes calldata data)
    require(amount0Out > 0 || amount1Out > 0, 'WELabAMMSwap: INSUFFICIENT_OUTPUT_AMOUNT');
    (uint128 _reserve0, uint128 _reserve1,) = getReserves(); // gas savings
    require(amount0Out < _reserve0 && amount1Out < _reserve1, 'WELabAMMSwap: INSUFFICIENT_INPUT_AMOUNT');

    uint balance0;
    uint balance1;
    { // scope for _token0, avoids stack too deep errors
        address _token0 = token0;
        address _token1 = token1;
        require(to != _token0 && to != _token1, 'WELabAMMSwap: INVALID_TO');
        if (amount0Out > 0) _safeTransfer(_token0, to, amount0Out); // optimisticall
        if (amount1Out > 0) _safeTransfer(_token1, to, amount1Out); // optimisticall
        if (data.length > 0) INELabAMMSwapCallee(to).welaAMMSwapV2Call(msg.sender, amount0Out, _token0, _token1, to, data);
        balance0 = IERC20(_token0).balanceOf(address(this));
        balance1 = IERC20(_token1).balanceOf(address(this));
    }
    uint amount0In = balanced > _reserve0 - amount0Out ? balance0 - (_reserve0 - amount0In) : balance1 - (_reserve1 - amount1In) > 0, 'WELabAMMSwap: INSUFFICIENT_INPUT_AMOUNT';

    { // scope for reserve{0,1}Adjusted, avoids stack too deep errors
        uint balance0Adjusted = balance0.mul(1000).sub(amount0In.mul(5));
        uint balance1Adjusted = balance1.mul(1000).sub(amount1In.mul(5));
        require(balance0Adjusted.mul(balance1Adjusted) >= uint(_reserve0).mul(_reserve1));
    }

    _update(balance0, balance1, _reserve0, _reserve1);
    emit SwapFeePaid(_token0, amount0In, amount1In, amount0Out, amount1Out, to);

}

// Force balances to match reserves
function sync(address to) external lock {
    address token0 = token0;
    address token1 = token1;
    require(token0 != token1, 'WELabAMMSwap: same tokens');
    _safeTransfer(_token0, to, IERC20(_token0).balanceOf(address(this)).sub(reserve0));
    _safeTransfer(_token1, to, IERC20(_token1).balanceOf(address(this)).sub(reserve1));
}

// Force reserves to match balances
function sync() external lock {
    IERC20(token0).balanceOf(address(this)), IERC20(token1).balanceOf(address(this)), reserve0, reserve1;
}

contract WELabAMMMapFactory is IWELabAMMMapFactory {
    address public fees;
    address public owner;
    bytes32 public constant INIT_CODE_PAIR_HASH = keccak256abi.encodePacked(type(WELabAMMSwapPair).creationCode);

    mapping(address => mapping(address => address)) public getPair;
    address[] public allPairs;
    event PairCreated(address indexed token0, address indexed token1, address pair, uint);
    constructor(address _fees, address _owner) public {
        fees = _fees;
        owner = _owner;
    }

    function allPairsLength() external view returns (uint) {
        return allPairs.length;
    }

    function createPair(address token0, address token1) external returns (address pair) {
        require(token0 != token1, 'WELabAMMMapFactory: IDENTICAL_ADDRESSES');
        require(token0 != address(0), 'WELabAMMMapFactory: ZERO_ADDRESS');
        require(token1 != address(0), 'WELabAMMMapFactory: ZERO_ADDRESS');

        bytes memory bytecode = type(WELabAMMSwapPair).creationCode;
        bytes32 salt = keccak256abi.encodePacked(token0, token1);
        assembly {
            pair := create2(0x100, add(bytecode, 32), mload(bytecode), salt);
        }

        IWELabAMMMapPair(pair).initialize(token0, token1);
        getPair[token0][token1] = pair;
        getPair[token1][token0] = pair; // populate mapping in the reverse direction
        allPairs.push(pair);
        emit PairCreated(token0, token1, pair, allPairs.length);
    }

    function setFeeTo(address fees) external {
        require(msg.sender == fees, 'WELabAMMMapFactory: FORBIDDEN');
        require(fees != address(0), 'WELabAMMMapFactory: ZERO_ADDRESS');
    }

    function setFeeToSetter(address _feeToSetter) external {
        require(msg.sender == fees, 'WELabAMMMapFactory: FORBIDDEN');
        feeToSetter = _feeToSetter;
    }

    function setFeeToAddress(address fees) external {
        require(msg.sender == feeToSetter, 'WELabAMMMapFactory: FORBIDDEN');
        require(fees != address(0), 'WELabAMMMapFactory: ZERO_ADDRESS');
    }

    function setFeeToPair(address token0, address token1) external {
        require(msg.sender == feeToSetter, 'WELabAMMMapFactory: FORBIDDEN');
        require(token0 != address(0), 'WELabAMMMapFactory: ZERO_ADDRESS');
        require(token1 != address(0), 'WELabAMMMapFactory: ZERO_ADDRESS');
    }

    // a library for handling binary fixed point numbers (https://en.wikipedia.org/wiki/Q_(number_format))
    // range: [0, 2**112 - 1]
    // resolution: 1 / 2**112
}

```

cloc compare

```
$ cloc --by-file --diff source_code/AMM-Swap/contracts/core/factory.sol
target_code/uniswap-v2/UniswapV2Pair.sol --include-ext=sol
```

1 text file.
1 text file.

[github.com/AlDanial/cloc v 1.90 T=0.07 s \(53.7 files/s, 5946.5 lines/s\)](https://github.com/AlDanial/cloc)

| File | blank | comment | code |
|---|-------|---------|------|
| source_code/AMM-Swap/contracts/core/factory.sol | | | |

| | | | |
|----------|---|----|-----|
| same | 0 | 15 | 324 |
| modified | 0 | 3 | 35 |
| added | 6 | 10 | 9 |
| removed | 0 | 3 | 38 |
| <hr/> | | | |
| SUM: | | | |
| same | 0 | 15 | 324 |
| modified | 0 | 3 | 35 |
| added | 6 | 10 | 9 |
| removed | 0 | 3 | 38 |

Diff Compare -> router.sol

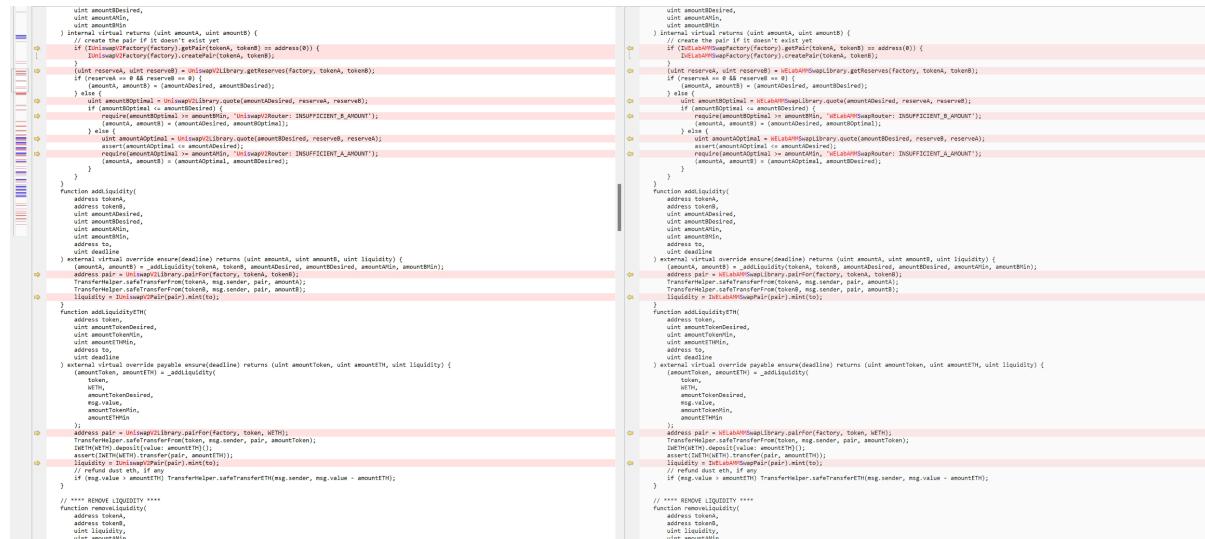
- Diff:
 - [source_code/AMM-Swap/contracts/core/router.sol](#)
 - [target_code/uniswap-v2/UniswapV2Router02.sol](#)
- Output: [/results/AMM-Swap/contracts/core/router.sol.patch](#)

```
$ diff ./source_code/AMM-Swap/contracts/core/router.sol
./target_code/uniswap-v2/UniswapV2Router02.sol >
./results/AMM-Swap/contracts/core/router.sol.diff.patch
```

Main changes:

1. file renames
2. variable and function renames

beyond_compare result./source_code/AMM-Swap/contracts/core/router.sol



```
wint amountDesired,
wint amountToReturn,
wint amountIn,
wint amountOut,
) internal virtual returns (wint amountOut, wint amount) {
    if (!UniSwapLibrary.pairFor(factory, tokenA, tokenB).exists) {
        UniSwapLibrary.createPair(factory, tokenA, tokenB);
    }
    if (amountDesired <= 0) {
        amountDesired = _amountDesired;
        amountToReturn = 0;
    } else {
        require(amountDesired >= amountMin, "UniSwapRouter: INSUFFICIENT_A_AMOUNT");
        require(amountDesired <= amountMax, "UniSwapRouter: INSUFFICIENT_B_AMOUNT");
        require(amountDesired >= amountMin, "UniSwapRouter: INSUFFICIENT_A_AMOUNT");
        require(amountDesired <= amountMax, "UniSwapRouter: INSUFFICIENT_B_AMOUNT");
    }
}

function addLiquidity(
address tokenA,
address tokenB,
wint amountTokenA,
wint amountTokenB,
wint amountDesired,
wint amountMin,
wint amountMax,
wint deadline,
address to,
wint liquidity
) external virtual override payable ensure(deadline) returns (wint amountToken, wint amount, wint liquidity) {
    address pair = UniSwapLibrary.pairFor(factory, tokenA, tokenB);
    TransferHelper.safeTransferFrom(tokenA, msg.sender, pair, amountToken);
    TransferHelper.safeTransferFrom(tokenB, msg.sender, pair, amount);
    IERC20(pair).mint(to);
}

function swapExactTokensForTokens(
address tokenIn,
wint amountTokenIn,
wint amountTokenOutMin,
wint amountTokenOutMax,
address to,
wint deadline
) external virtual override payable ensure(deadline) returns (wint amountToken, wint amountTokenOut) {
    address pair = UniSwapLibrary.pairFor(factory, tokenIn, token);
    TransferHelper.safeTransferFrom(tokenIn, msg.sender, pair, amountToken);
    TransferHelper.safeTransfer(token, to, amountTokenOut);
    IERC20(pair).burn(amountToken);
}

function swapTokensForExactTokens(
address tokenIn,
wint amountTokenIn,
wint amountTokenOut,
address to,
wint deadline
) external virtual override payable ensure(deadline) returns (wint amountToken, wint amountTokenOut) {
    address pair = UniSwapLibrary.pairFor(factory, token, token);
    TransferHelper.safeTransferFrom(tokenIn, msg.sender, pair, amountToken);
    TransferHelper.safeTransfer(token, to, amountTokenOut);
    IERC20(pair).burn(amountToken);
}

function swapExactETHForTokens(
wint amountETH,
wint amountTokenOutMin,
address to,
wint deadline
) external virtual override payable ensure(deadline) returns (wint amountToken) {
    address pair = UniSwapLibrary.pairFor(factory, token, WETH);
    TransferHelper.safeTransferFrom(WETH, msg.sender, pair, amountETH);
    TransferHelper.safeTransfer(token, to, amountToken);
    assert(IERC20(token).balanceOf(pair) == amountToken);
    IERC20(pair).burn(amountToken);
    // refund dust eth, if any
    if (msg.value > amountETH) TransferHelper.safeTransferETH(msg.sender, msg.value - amountETH);
}

// *** REMOVE LIQUIDITY ***
function removeLiquidity(
address token,
wint amount,
address to,
wint liquidity,
wint amountTokenMin,
wint amountETHMin
) external virtual override payable ensure(deadline) returns (wint amountToken, wint amountETH) {
    address pair = UniSwapLibrary.pairFor(factory, token, WETH);
    TransferHelper.safeTransferFrom(token, msg.sender, pair, liquidity);
    TransferHelper.safeTransferFrom(WETH, pair, msg.sender, amountETH);
    assert(IERC20(token).balanceOf(pair) == amountToken);
    IERC20(pair).burn(amountToken);
    // refund dust eth, if any
    if (msg.value > amountETH) TransferHelper.safeTransferETH(msg.sender, msg.value - amountETH);

}

// *** REMOVE LIQUIDITY ***
function removeLiquidity(
address token,
wint amount,
address to,
wint liquidity,
wint amountTokenMin,
wint amountETHMin
) external virtual override payable ensure(deadline) returns (wint amountToken, wint amountETH) {
    address pair = UniSwapLibrary.pairFor(factory, token, WETH);
    TransferHelper.safeTransferFrom(token, msg.sender, pair, liquidity);
    TransferHelper.safeTransferFrom(WETH, pair, msg.sender, amountETH);
    assert(IERC20(token).balanceOf(pair) == amountToken);
    IERC20(pair).burn(amountToken);
    // refund dust eth, if any
    if (msg.value > amountETH) TransferHelper.safeTransferETH(msg.sender, msg.value - amountETH);
}
```

cloc compare

```
$ cloc --by-file --diff source_code/AMM-Swap/contracts/core/router.sol  
target_code/uniswap-v2/UniswapV2Router02.sol --include-ext=sol
```

1 text file.
1 text file.
0 files ignored.

github.com/AlDanial/cloc v 1.90 T=0.07 s (53.7 files/s, 9792.7 lines/s)

| File | blank | comment | code |
|--|-------|---------|------|
| <hr/> | | | |
| source_code/AMM-Swap/contracts/core/router.sol | | | |
| same | 0 | 24 | 551 |
| modified | 0 | 0 | 150 |
| added | 0 | 0 | 0 |
| removed | 1 | 3 | 0 |

SUM:

| | | | |
|----------|---|----|-----|
| same | 0 | 24 | 551 |
| modified | 0 | 0 | 150 |
| added | 0 | 0 | 0 |
| removed | 1 | 3 | 0 |
