KwikSwap - Factory Contract

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **May 31, 2021 - June 10, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY		4
CONTACTS		4
1	EXECUTIVE OVERVIEW	5
1.1	INTRODUCTION	6
1.2	AUDIT SUMMARY	6
1.3	TEST APPROACH & METHODOLOGY	7
	RISK METHODOLOGY	7
1.4	SCOPE	9
2	ASSESSMENT SUMMARY & FINDINGS OVERVIEW	10
3	FINDINGS & TECH DETAILS	11
3.1	(HAL-01) UNCHECKED TRANSFER - MEDIUM	13
	Description	13
	Code Location	13
	Risk Level	14
	Recommendation	14
	Remediation Plan	14
3.2	(HAL-02) PRAGMA VERSION DEPRECATED - LOW	15
	Description	15
	Code Location	15
	Risk Level	15
	Recommendations	15
	Remediation Plan	16
3.3	(HAL-03) USE OF BLOCK.TIMESTAMP - LOW	17
	Description	17

Code Location 17
Risk Level 18
Recommendation 18
Remediation Plan 18
3.4 (HAL-04) MISSING ZERO-ADDRESS CHECK - LOW 19
Description 19
Code Location 19
Risk Level 20
Recommendation 20
Remediation Plan 20
3.5 (HAL-05) IGNORE RETURN VALUES - LOW 21
Description 21
Code Location 21
Risk Level 21
Recommendation 22
Remediation Plan 22
3.6 (HAL-06) MISSING RE-ENTRANCY PROTECTION - INFORMATIONAL 23
Description 23
Risk Level 23
Recommendation 23
Remediation Plan 23
4 AUTOMATED TESTING 24
4.1 STATIC ANALYSIS REPORT 25
Description 25
Result 25
4.2 AUTOMATED SECURITY SCAN 28
MYTHX 28

Results 28

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	05/30/2021	Gabi Urrutia
0.2	Document Edits	06/02/2021	Nishit Majithia
1.0	Final Draft	06/10/2021	Nishit Majithia
1.1	Final Version	06/29/2021	Nishit Majithia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Nishit Majithia	Halborn	nishit.majithia@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Kwikswap is a Multi Cross-Chain Swap Protocol with Layer 2 Scaling powered by Ethereum, Polkadot, Plasm, Reef Chain, BSC & Acala Network. KwikSwap is a decentralised protocol built primarily on the Ethereum Network. KwikSwap allows the creation of token markets, own KWIK token, Non-Custodial wallet connection, no need for KYC, features layer 2 scaling and you always control your funds for a completely decentralized experience.

Kwikswap engaged Halborn to conduct a security assessment on their Smart contracts on May 31st, 2021 to June 10th, 2021. The security assessment was scoped to the smart contract provided in the Github repository Kwikswap Smart Contracts and an audit of the security risk and implications regarding the changes introduced by the development team at Kwikswap prior to its production release shortly following the assessments deadline.

Though this security audit's outcome is satisfactory, only the most essential aspects were tested and verified to achieve objectives and deliverables set in the scope due to time and resource constraints. It is essential to note the use of the best practices for secure smart-contract development.

1.2 AUDIT SUMMARY

The team at Halborn was provided a week for the engagement and assigned a full time security engineer to audit the security of the smart contract. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit to achieve the following:

- Ensure that smart contract functions are intended.
- Identify potential security issues with the smart contracts.

In summary, Halborn identified few security risks, and recommends performing further testing to validate extended safety and correctness in context to the whole set of contracts. External threats, such as economic attacks, oracle attacks, and inter-contract functions and calls should be validated for expected logic and state.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the smart contract audit. While manual testing is recommended to uncover flaws in logic, process,and implementation; automated testing techniques help enhance coverage of smart contracts and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose.
- Smart Contract manual code review and walkthrough.
- Graphing out functionality and contract logic/connectivity/functions(solgraph)
- Manual Assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes (brownie console and manual deployments on Ganache)
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions.(Slither)
- Testnet deployment (Ganache, Remix IDE)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident, and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. It's quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that was used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

IN-SCOPE:

The security assessment was scoped to the smart contracts:

- Kiwkswap Router02.sol
- Kwikswap Router01.sol
- KwikswapV1ERC20.sol
- KwikswapV1Factory.sol
- KwikswapV1Pair.sol
- Contracts unders libraries/ and interfaces/ directories

Commit ID: f4e2f0481462b7bc40e5c4d94768a8acc8cf7d22

Fix Commit ID: 66fce626b57908af56301e1e6c84f672c55a0298

OUT-OF-SCOPE:

External libraries and economics attacks.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	1	4	1

LIKELIHOOD

IMPACT

(HAL-03)				
(HAL-02)		(HAL-01)		
	(HAL-04)	(HAL-05)		
(HAL-06)				

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
UNCHECKED TRANSFER	Medium	SOLVED - 06/29/2021
PRAGMA VERSION DEPRECATED	Low	RISK ACCEPTED
USE OF BLOCK.TIMESTAMP	Low	SOLVED - 06/29/2021
MISSING ZERO-ADDRESS CHECKS	Low	SOLVED - 06/29/2021
IGNORE RETURN VALUES	Low	SOLVED - 06/29/2021
MISSING RE-ENTRANCY PROTECTION	Informational	RISK ACCEPTED

FINDINGS & TECH DETAILS

3.1 (HAL-01) UNCHECKED TRANSFER - MEDIUM

Description:

The contracts Kiwkswap Router02.sol and Kwikswap Router01.sol has removeLiquidity() method and in this method, transferFrom() is being called without any implementing checks on the return value. Several tokens do not revert in case of failure and return false.

Code Location:

Listing 1: Kiwkswap Router02.sol (Lines 484)

```
483        address pair = KwikswapV1Library.pairFor(factory, tokenA, tokenB);
484        IKwikswapV1Pair(pair).transferFrom(msg.sender, pair, liquidity); // send liquidity to pair
485        (uint amount0, uint amount1) = IKwikswapV1Pair(pair).burn(to);
486        (address token0,) = KwikswapV1Library.sortTokens(tokenA, tokenB);
```

Listing 2: Kwikswap Router01.sol (Lines 466)

```
465        address pair = KwikswapV1Library.pairFor(factory, tokenA, tokenB);
466        IKwikswapV1Pair(pair).transferFrom(msg.sender, pair, liquidity); // send liquidity to pair
467        (uint amount0, uint amount1) = IKwikswapV1Pair(pair).burn(to);
468        (address token0,) = KwikswapV1Library.sortTokens(tokenA, tokenB);
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

Use SafeERC20, or ensure that the transferFrom return value is checked.

Remediation Plan:

SOLVED: Contracts Kiwkswap Router02 and Kiwkswap Router01 are now using SafeERC20.

3.2 (HAL-02) PRAGMA VERSION DEPRECATED - LOW

Description:

The current version in use for the contracts is pragma =0.5.12 for KwikswapV1ERC20.sol, KwikswapV1Factory.sol and KwikswapV1Pair.sol while =0.6.6 for Kiwkswap Router02.sol and Kwikswap Router01.sol. While these version is still functional, and most security issues safely implemented by mitigating contracts with other utility contracts such as SafeMath.sol and ReentrancyGuard.sol, the risk to the long-term sustainability and integrity of the solidity code increases.

Code Location:

Listing 3: KwikswapV1ERC20.sol, KwikswapV1Factory.sol and KwikswapV1Pair.sol

```
pragma solidity =0.5.16;
```

Listing 4: Kiwkswap Router02.sol and Kwikswap Router01.sol

```
pragma solidity =0.6.6;
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendations:

At the time of this audit, the current version is already at 0.8.2. When possible, use the most updated and tested pragma versions to take advantage of new features that provide checks and accounting, as well as prevent insecure use of code. (0.6.12)

Remediation Plan:

RISK ACCEPTED: Kwikswap team wants to continue with this pragma version.

3.3 (HAL-03) USE OF BLOCK.TIMESTAMP - LOW

Description:

During a manual static review, the tester noticed the use of block.timestamp in KwikswapV1ERC20.sol and KwikswapV1Pair.sol contracts. The contract developers should be aware that this does not mean current time. Miners can influence the value of block.timestamp to perform Maximal Extractable Value (MEV) attacks. The use of now creates a risk that time manipulation can be performed to manipulate price oracles. Miners can modify the timestamp by up to 900 seconds.

Code Location:

Listing 5: KwikswapV1ERC20.sol (Lines 82)

```
81     function permit(address owner, address spender, uint value,
           uint deadline, uint8 v, bytes32 r, bytes32 s) external {
82         require(deadline >= block.timestamp, 'KwikswapV1: EXPIRED'
               );
83         bytes32 digest = keccak256(
84             abi.encodePacked(
85                 '\x19\x01',
86                 DOMAIN_SEPARATOR,
87                 keccak256(abi.encode(PERMIT_TYPEHASH, owner,
                       spender, value, nonces[owner]++, deadline))
88             )
```

Listing 6: KwikswapV1Pair.sol (Lines 77)

```
73     function _update(uint balance0, uint balance1, uint112
           _reserve0, uint112 _reserve1) private {
74         require(balance0 <= uint112(-1) && balance1 <= uint112(-1)
               , 'KwikswapV1: OVERFLOW');
75         uint32 blockTimestamp = uint32(block.timestamp % 2**32);
76         uint32 timeElapsed = blockTimestamp - blockTimestampLast;
               // overflow is desired
77         if (timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0) {
```

Risk Level:

Likelihood - 1
Impact - 4

Recommendation:

Use `block.number` instead of `block.timestamp` to reduce the risk of MEV attacks. Check if the timescale of the project occurs across years, days and months rather than seconds. If possible, it is recommended to use Oracles.

Remediation Plan:

SOLVED: Contracts `KwikswapV1Pair` and `KwikswapV1ERC20` are now using `block.number` instead of `block.timestamp`.

3.4 (HAL-04) MISSING ZERO-ADDRESS CHECK - LOW

Description:

The contracts KwikswapV1Factory.sol, KwikswapV1Pair.sol, Kiwkswap Router02.sol and Kwikswap Router01.sol should perform a “zero-address” validation check when assigning the user supplied address values to state variables.

Code Location:

Listing 7: KwikswapV1Factory.sol (Lines 16)

```
15     constructor(address _feeToSetter) public {
16         feeToSetter = _feeToSetter;
17     }
```

Listing 8: KwikswapV1Factory.sol (Lines 42)

```
40     function setFeeTo(address _feeTo) external {
41         require(msg.sender == feeToSetter, 'KwikswapV1: FORBIDDEN'
               );
42         feeTo = _feeTo;
43     }
```

Listing 9: KwikswapV1Factory.sol (Lines 47)

```
45     function setFeeToSetter(address _feeToSetter) external {
46         require(msg.sender == feeToSetter, 'KwikswapV1: FORBIDDEN'
               );
47         feeToSetter = _feeToSetter;
48     }
```

Listing 10: KwikswapV1Pair.sol (Lines 68,69)

```
66    function initialize(address _token0, address _token1) external
          {
67        require(msg.sender == factory, 'KwikswapV1: FORBIDDEN');
              // sufficient check
68        token0 = _token0;
69        token1 = _token1;
70    }
```

Listing 11: Kiwkswap Router02.sol (Lines 395,396)

```
394   constructor(address _factory, address _WETH) public {
395       factory = _factory;
396       WETH = _WETH;
397   }
```

Listing 12: Kwikswap Router01.sol (Lines 378,379)

```
377   constructor(address _factory, address _WETH) public {
378       factory = _factory;
379       WETH = _WETH;
380   }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

Add address validation for user-supplied values in addition to the existing OpenZeppelin RBAC controls.

Remediation Plan:

SOLVED: Contracts KwikswapV1Pair, KwikswapV1Factory, Kwikswap Router01 and Kiwkswap Router02 added address validation for user-supplied values.

3.5 (HAL-05) IGNORE RETURN VALUES - LOW

Description:

The return value of an external call is not stored in a local or state variable. In the contracts Kiwkswap Router02.sol and Kwikswap Router01.sol, there are instances where external methods are being called and the return value is being ignored.

Code Location:

Listing 13: Kiwkswap Router02.sol (Lines 414)

```
413         if (IKwikswapV1Factory(factory).getPair(tokenA, tokenB) ==
                address(0)) {
414             IKwikswapV1Factory(factory).createPair(tokenA, tokenB)
                    ;
415         }
416         (uint reserveA, uint reserveB) = KwikswapV1Library.
                getReserves(factory, tokenA, tokenB);
```

Listing 14: Kwikswap Router01.sol (Lines 397)

```
396         if (IKwikswapV1Factory(factory).getPair(tokenA, tokenB) ==
                address(0)) {
397             IKwikswapV1Factory(factory).createPair(tokenA, tokenB)
                    ;
398         }
399         (uint reserveA, uint reserveB) = KwikswapV1Library.
                getReserves(factory, tokenA, tokenB);
```

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

Add a return value check to avoid unexpected errors. Return value checks ensure proper exception handling.

Remediation Plan:

SOLVED: Contracts Kwikswap Router01 and Kwikswap Router02 added a return value check to avoid unexpected errors.

3.6 (HAL-06) MISSING RE-ENTRANCY PROTECTION - INFORMATIONAL

Description:

To protect against cross-function reentrancy attacks, it may be necessary to use a mutex. By using this lock, an attacker can no longer exploit the withdraw function with a recursive call. OpenZeppelin has it's own mutex implementation called ReentrancyGuard which provides a modifier to any function called nonReentrant that guards the function with a mutex against reentrancy attacks.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

In the KwikswapV1Pair.sol and KwikswapV1Factory.sol contracts, functions like KwikswapV1Pair.burn(), KwikswapV1Pair.swap() and KwikswapV1Factory.createPair() are missing nonReentrant guard. Use the nonReentrant modifier to avoid introducing future vulnerabilities.

Remediation Plan:

RISK ACCEPTED: Kwikswap team are fine with not adding nonReenrant guard in their contracts.

AUTOMATED TESTING

4.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance coverage of certain areas of the scoped contract. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their abi and binary formats. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Result:

```
INFO:Detectors:
KwikswapV1Pair._update(uint256,uint256,uint112,uint112) (contracts/KwikswapV1Pair.sol#73-86) uses a weak PRNG: "blockTimestamp = uint32(block.timestamp % 2 ** 32) (contracts/KwikswapV1Pair.sol#75)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG
```

- This issue is false positive since there is no situation here for 'weak PRNG', the code is just trying to convert block.timestamp value to uint32. It is not using it as a source of randomness.

```
INFO:Detectors:
KwikswapV1Router02.removeLiquidity(address,address,uint256,uint256,uint256,address,uint256) (contracts/Kiwkswap Router02.sol#474-490) ignores return value by IKwikswapV1Pair(pair).transferFrom(msg.sender,pair,liquidi
ty) (contracts/Kiwkswap Router02.sol#484)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
KwikswapV1Router01.removeLiquidity(address,address,uint256,uint256,uint256,address,uint256) (contracts/Kwikswap Router01.sol#456-472) ignores return value by IKwikswapV1Pair(pair).transferFrom(msg.sender,pair,liquidi
ty) (contracts/Kwikswap Router01.sol#466)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
```

- This 'unchecked transfer' issue is already reported in the above report.

```
INFO:Detectors:
Reentrancy in KwikswapV1Pair.burn(address) (contracts/KwikswapV1Pair.sol#134-156):
        External calls:
        - _safeTransfer(_token0,to,amount0) (contracts/KwikswapV1Pair.sol#148)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/KwikswapV1Pair.sol#45)
        - _safeTransfer(_token1,to,amount1) (contracts/KwikswapV1Pair.sol#149)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/KwikswapV1Pair.sol#45)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/KwikswapV1Pair.sol#153)
                - blockTimestampLast = blockTimestamp (contracts/KwikswapV1Pair.sol#84)
        - kLast = uint256(reserve0).mul(reserve1) (contracts/KwikswapV1Pair.sol#154)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/KwikswapV1Pair.sol#153)
                - reserve0 = uint112(balance0) (contracts/KwikswapV1Pair.sol#82)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/KwikswapV1Pair.sol#153)
                - reserve1 = uint112(balance1) (contracts/KwikswapV1Pair.sol#83)
Reentrancy in KwikswapV1Factory.createPair(address,address) (contracts/KwikswapV1Factory.sol#23-38):
        External calls:
        - IKwikswapV1Pair(pair).initialize(token0,token1) (contracts/KwikswapV1Factory.sol#33)
        State variables written after the call(s):
        - getPair[token0][token1] = pair (contracts/KwikswapV1Factory.sol#34)
        - getPair[token1][token0] = pair (contracts/KwikswapV1Factory.sol#35)
Reentrancy in KwikswapV1Pair.swap(uint256,uint256,address,bytes) (contracts/KwikswapV1Pair.sol#159-187):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (contracts/KwikswapV1Pair.sol#170)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/KwikswapV1Pair.sol#45)
        - _safeTransfer(_token1,to,amount1Out) (contracts/KwikswapV1Pair.sol#171)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/KwikswapV1Pair.sol#45)
        - IKwikswapV1Callee(to).kwikswapV1Call(msg.sender,amount0Out,amount1Out,data) (contracts/KwikswapV1Pair.sol#172)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/KwikswapV1Pair.sol#185)
                - blockTimestampLast = blockTimestamp (contracts/KwikswapV1Pair.sol#84)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/KwikswapV1Pair.sol#185)
                - reserve0 = uint112(balance0) (contracts/KwikswapV1Pair.sol#82)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/KwikswapV1Pair.sol#185)
                - reserve1 = uint112(balance1) (contracts/KwikswapV1Pair.sol#83)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
```

- Re-entrancy issue is not present but in future to protect against cross-function reentrancy attacks, it may be necessary to use a mutex. By using this lock, an attacker can no longer exploit the withdraw function with a recursive call. OpenZeppelin has it's own mutex implementation called ReentrancyGuard which provides a modifier to any function called nonReentrant that guards the function with a mutex against reentrancy attacks.

```
INFO:Detectors:
KwikswapV1Router02._addLiquidity(address,address,uint256,uint256,uint256,uint256) (contracts/Kiwkswap Router02.sol#404-431) ignores return value by IKwikswapV1Factory(factory).createPair(tokenA,tokenB) (contracts/Kiwkswap Router02.sol#414)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
KwikswapV1Router01._addLiquidity(address,address,uint256,uint256,uint256,uint256) (contracts/Kwikswap Router01.sol#387-414) ignores return value by IKwikswapV1Factory(factory).createPair(tokenA,tokenB) (contracts/Kwikswap Router01.sol#397)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

- All unused return issues are present and it is good to add a return value check to avoid unexpected errors. Return value checks ensure proper exception handling.

```
INFO:Detectors:
KwikswapV1Router02._swapSupportingFeeOnTransferTokens(address[],address).i (contracts/Kiwkswap Router02.sol#693) is a local variable never initialized
KwikswapV1Router02._swap(uint256[],address[],address).i (contracts/Kiwkswap Router02.sol#584) is a local variable never initialized
KwikswapV1Library.getAmountsOut(address,uint256,address[]).i (contracts/Kiwkswap Router02.sol#128) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
KwikswapV1Router01._swap(uint256[],address[],address).i (contracts/Kwikswap Router01.sol#527) is a local variable never initialized
KwikswapV1Library.getAmountsOut(address,uint256,address[]).i (contracts/Kwikswap Router01.sol#159) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
```

- All these 'uninitialized variable issues are false positive since variable i is being used in for loop and since it is uint it’s default value is zero.

```
INFO:Detectors:
KwikswapV1ERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (contracts/KwikswapV1ERC20.sol#81-93) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,KwikswapV1: EXPIRED) (contracts/KwikswapV1ERC20.sol#82)
KwikswapV1Pair._update(uint256,uint256,uint112,uint112) (contracts/KwikswapV1Pair.sol#73-86) uses timestamp for comparisons
        Dangerous comparisons:
        - timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0 (contracts/KwikswapV1Pair.sol#77)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

- Issues regarding 'block.timestamp' has been already raised in the report above.

```
INFO:Detectors:
quote(uint256,uint256,uint256) should be declared external:
        - KwikswapV1Router02.quote(uint256,uint256,uint256) (contracts/Kiwkswap Router02.sol#774-776)
getAmountOut(uint256,uint256,uint256) should be declared external:
        - KwikswapV1Router02.getAmountOut(uint256,uint256,uint256) (contracts/Kiwkswap Router02.sol#778-786)
getAmountIn(uint256,uint256,uint256) should be declared external:
        - KwikswapV1Router02.getAmountIn(uint256,uint256,uint256) (contracts/Kiwkswap Router02.sol#788-796)
getAmountsOut(uint256,address[]) should be declared external:
        - KwikswapV1Router02.getAmountsOut(uint256,address[]) (contracts/Kiwkswap Router02.sol#798-806)
getAmountsIn(uint256,address[]) should be declared external:
        - KwikswapV1Router02.getAmountsIn(uint256,address[]) (contracts/Kiwkswap Router02.sol#808-816)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Detectors:
quote(uint256,uint256,uint256) should be declared external:
        - KwikswapV1Router01.quote(uint256,uint256,uint256) (contracts/Kwikswap Router01.sol#618-620)
getAmountOut(uint256,uint256,uint256) should be declared external:
        - KwikswapV1Router01.getAmountOut(uint256,uint256,uint256) (contracts/Kwikswap Router01.sol#622-624)
getAmountIn(uint256,uint256,uint256) should be declared external:
        - KwikswapV1Router01.getAmountIn(uint256,uint256,uint256) (contracts/Kwikswap Router01.sol#626-628)
getAmountsOut(uint256,address[]) should be declared external:
        - KwikswapV1Router01.getAmountsOut(uint256,address[]) (contracts/Kwikswap Router01.sol#630-632)
getAmountsIn(uint256,address[]) should be declared external:
        - KwikswapV1Router01.getAmountsIn(uint256,address[]) (contracts/Kwikswap Router01.sol#634-636)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

- Issues regarding 'function visibility' has been already raised in the report above.

```
INFO:Detectors:
Reentrancy in KwikswapV1Pair.burn(address) (contracts/KwikswapV1Pair.sol#134-156):
        External calls:
        - _safeTransfer(_token0,to,amount0) (contracts/KwikswapV1Pair.sol#148)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/KwikswapV1Pair.sol#45)
        - _safeTransfer(_token1,to,amount1) (contracts/KwikswapV1Pair.sol#149)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/KwikswapV1Pair.sol#45)
        Event emitted after the call(s):
        - Burn(msg.sender,amount0,amount1,to) (contracts/KwikswapV1Pair.sol#155)
        - Sync(reserve0,reserve1) (contracts/KwikswapV1Pair.sol#85)
                - _update(balance0,balance1,_reserve0,_reserve1) (contracts/KwikswapV1Pair.sol#153)
Reentrancy in KwikswapV1Factory.createPair(address,address) (contracts/KwikswapV1Factory.sol#23-38):
        External calls:
        - IKwikswapV1Pair(pair).initialize(token0,token1) (contracts/KwikswapV1Factory.sol#33)
        Event emitted after the call(s):
        - PairCreated(token0,token1,pair,allPairs.length) (contracts/KwikswapV1Factory.sol#37)
Reentrancy in KwikswapV1Pair.swap(uint256,uint256,address,bytes) (contracts/KwikswapV1Pair.sol#159-187):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (contracts/KwikswapV1Pair.sol#170)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/KwikswapV1Pair.sol#45)
        - _safeTransfer(_token1,to,amount1Out) (contracts/KwikswapV1Pair.sol#171)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/KwikswapV1Pair.sol#45)
        - IKwikswapV1Callee(to).kwikswapV1Call(msg.sender,amount0Out,amount1Out,data) (contracts/KwikswapV1Pair.sol#172)
        Event emitted after the call(s):
        - Swap(msg.sender,amount0In,amount1In,amount0Out,amount1Out,to) (contracts/KwikswapV1Pair.sol#186)
        - Sync(reserve0,reserve1) (contracts/KwikswapV1Pair.sol#85)
                - _update(balance0,balance1,_reserve0,_reserve1) (contracts/KwikswapV1Pair.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
INFO:Detectors:
Reentrancy in KwikswapV1Pair.burn(address) (contracts/KwikswapV1Pair.sol#134-156):
        External calls:
        - _safeTransfer(_token0,to,amount0) (contracts/KwikswapV1Pair.sol#148)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/KwikswapV1Pair.sol#45)
        - _safeTransfer(_token1,to,amount1) (contracts/KwikswapV1Pair.sol#149)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/KwikswapV1Pair.sol#45)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/KwikswapV1Pair.sol#153)
                - price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (contracts/KwikswapV1Pair.sol#79)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/KwikswapV1Pair.sol#153)
                - price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (contracts/KwikswapV1Pair.sol#80)
Reentrancy in KwikswapV1Factory.createPair(address,address) (contracts/KwikswapV1Factory.sol#23-38):
        External calls:
        - IKwikswapV1Pair(pair).initialize(token0,token1) (contracts/KwikswapV1Factory.sol#33)
        State variables written after the call(s):
        - allPairs.push(pair) (contracts/KwikswapV1Factory.sol#36)
Reentrancy in KwikswapV1Pair.swap(uint256,uint256,address,bytes) (contracts/KwikswapV1Pair.sol#159-187):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (contracts/KwikswapV1Pair.sol#170)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/KwikswapV1Pair.sol#45)
        - _safeTransfer(_token1,to,amount1Out) (contracts/KwikswapV1Pair.sol#171)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/KwikswapV1Pair.sol#45)
        - IKwikswapV1Callee(to).kwikswapV1Call(msg.sender,amount0Out,amount1Out,data) (contracts/KwikswapV1Pair.sol#172)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/KwikswapV1Pair.sol#185)
                - price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (contracts/KwikswapV1Pair.sol#79)
        - _update(balance0,balance1,_reserve0,_reserve1) (contracts/KwikswapV1Pair.sol#185)
                - price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (contracts/KwikswapV1Pair.sol#80)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

- Re-entrancy issue is not present but in future to protect against cross-function reentrancy attacks, it may be necessary to use a mutex. By using this lock, an attacker can no longer exploit the withdraw function with a recursive call. OpenZeppelin has it's own mutex implementation called ReentrancyGuard which provides a modifier to any function called nonReentrant that guards the function with a mutex against reentrancy attacks.

```
INFO:Detectors:
KwikswapV1Factory.constructor(address)._feeToSetter (contracts/KwikswapV1Factory.sol#15) lacks a zero-check on :
                - feeToSetter = _feeToSetter (contracts/KwikswapV1Factory.sol#16)
KwikswapV1Factory.setFeeTo(address)._feeTo (contracts/KwikswapV1Factory.sol#40) lacks a zero-check on :
                - feeTo = _feeTo (contracts/KwikswapV1Factory.sol#42)
KwikswapV1Factory.setFeeToSetter(address)._feeToSetter (contracts/KwikswapV1Factory.sol#45) lacks a zero-check on :
                - feeToSetter = _feeToSetter (contracts/KwikswapV1Factory.sol#47)
KwikswapV1Pair.initialize(address,address)._token0 (contracts/KwikswapV1Pair.sol#66) lacks a zero-check on :
                - token0 = _token0 (contracts/KwikswapV1Pair.sol#68)
KwikswapV1Pair.initialize(address,address)._token1 (contracts/KwikswapV1Pair.sol#66) lacks a zero-check on :
                - token1 = _token1 (contracts/KwikswapV1Pair.sol#69)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
INFO:Detectors:
KwikswapV1Router02.constructor(address,address)._factory (contracts/Kiwkswap Router02.sol#394) lacks a zero-check on :
                - factory = _factory (contracts/Kiwkswap Router02.sol#395)
KwikswapV1Router02.constructor(address,address)._WETH (contracts/Kiwkswap Router02.sol#394) lacks a zero-check on :
                - WETH = _WETH (contracts/Kiwkswap Router02.sol#396)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
INFO:Detectors:
KwikswapV1Router01.constructor(address,address)._factory (contracts/Kwikswap Router01.sol#377) lacks a zero-check on :
                - factory = _factory (contracts/Kwikswap Router01.sol#378)
KwikswapV1Router01.constructor(address,address)._WETH (contracts/Kwikswap Router01.sol#377) lacks a zero-check on :
                - WETH = _WETH (contracts/Kwikswap Router01.sol#379)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

- Issues regarding 'missing zero address validation' has been already raised in the report above.

4.2 AUTOMATED SECURITY SCAN

MYTHX:

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruit on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on the testers machine and sent the compiled results to the analyzers to locate any vulnerabilities. Only security-related findings are shown below.

Results:

Kwikswap Router01.sol

Report for contracts/Kwikswap Router01.sol
https://dashboard.mythx.io/#/console/analyses/fec0c805-b966-4b76-ad4c-1114f34df386

Line	SWC Title	Severity	Short Description
47	(SWC-103) Floating Pragma	Low	A floating pragma is set.
78	(SWC-103) Floating Pragma	Low	A floating pragma is set.
98	(SWC-103) Floating Pragma	Low	A floating pragma is set.
178	(SWC-103) Floating Pragma	Low	A floating pragma is set.
200	(SWC-103) Floating Pragma	Low	A floating pragma is set.
257	(SWC-103) Floating Pragma	Low	A floating pragma is set.
356	(SWC-103) Floating Pragma	Low	A floating pragma is set.
383	(SWC-110) Assert Violation	Low	An assertion violation was triggered.

Kwikswap Router02.sol

Report for contracts/Kiwkswap Router02.sol
https://dashboard.mythx.io/#/console/analyses/369eef4c-7b82-47e1-957f-7f297eecca9f

Line	SWC Title	Severity	Short Description
120	(SWC-110) Assert Violation	Low	An assertion violation was triggered.
400	(SWC-110) Assert Violation	Low	An assertion violation was triggered.

KwikswapV1ERC20.sol

Report for KwikswapV1ERC20.sol
https://dashboard.mythx.io/#/console/analyses/15b1ffb0-3864-4d00-a3ba-e370be312268

Line	SWC Title	Severity	Short Description
82	(SWC-116) Timestamp Dependence	Low	A control flow decision is made based on The block.timestamp environment variable.

KwikswapV1Pair.sol

Report for KwikswapV1Pair.sol
https://dashboard.mythx.io/#/console/analyses/5198b672-29e7-413f-85c3-5029657962e4

Line	SWC Title	Severity	Short Description
16	(SWC-128) DoS With Block Gas Limit	Low	Potentially unbounded data structure passed to builtin.
35	(SWC-107) Reentrancy	Low	Write to persistent state following external call
45	(SWC-113) DoS with Failed Call	Low	Multiple calls are executed in the same transaction.
77	(SWC-116) Timestamp Dependence	Low	A control flow decision is made based on The block.timestamp environment variable.
194	(SWC-107) Reentrancy	Low	Read of persistent state following external call
199	(SWC-113) DoS with Failed Call	Low	Multiple calls are executed in the same transaction.

- Issues regarding 'floating pragma' and 'block.timestamp' has been already raised in the report above.

THANK YOU FOR CHOOSING

HALBORN