FAIRYPROOF

Zebra V3

AUDIT REPORT

Version 1.0.0

Serial No. 2023112200012016

Presented by Fairyproof

November 22, 2023

01. Introduction

This document includes the results of the audit performed by the Fairyproof team on the Zebra decentralized exchange project.

Audit Start Time:

November 17, 2023

Audit End Time:

November 22, 2023

Audited Source File's Address:

ZebraV3Factory:
https://scrollscan.com/address/0x96a7F53f7636c93735bf85dE416A4Ace94B56Bd9#code
QuoterV2:
https://scrollscan.com/address/0xbC92FAfA262458F05986B2F7B1056c21f812ba48#code
SwapRouter:
https://scrollscan.com/address/0x5A4c258a6c7a6a6816eC6a71FF2187D20178781d#code
TickLens:
https://scrollscan.com/address/0x1Daef3F1D97CcD9ae428641B23733cF1C619AeAA#code
NonfungiblePositionManager:
https://scrollscan.com/address/0x1b009B28BB2cD55Ec5CCedAE2E6dC9d45a25eF2f#code
V3Migrator:
https://scrollscan.com/address/0x3f123B652290e435035860feC34D983FDfC5e1DF#code
ZebraInterfaceMulticall:
https://scrollscan.com/address/0x22EdB96DeE65847502e0908604FA5DfA7284A68b#code

Audited Code's Github Repository:

https://github.com/zebra-xyz/v3-contracts

https://github.com/zebra-xyz/universal-router

Audited Code's Github Commit Number When Audit Started:

zebra-v3: d345ef9efc8eff8b07e03fde80cb67fdcded67b2

universal-router: ad9f84c72be17b8630a00095524cfa1b3748d328

The goal of this audit is to review Zebra’s solidity implementation for its Decentralized exchange function, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

We make observations on specific areas of the code that present concrete problems, as well as general observations that traverse the entire codebase horizontally, which could improve its quality as a whole.

This audit only applies to the specified code, software or any materials supplied by the Zebra team for specified versions. Whenever the code, software, materials, settings, environment etc is changed, the comments of this audit will no longer apply.

— Disclaimer

Note that as of the date of publishing, the contents of this report reflect the current understanding of known security patterns and state of the art regarding system security. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk.

The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. If the audited source files are smart contract files, risks or issues introduced by using data feeds from offchain sources are not extended by this review either.

Given the size of the project, the findings detailed here are not to be considered exhaustive, and further testing and audit is recommended after the issues covered are fixed.

To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

— Methodology

The above files' code was studied in detail in order to acquire a clear impression of how the its specifications were implemented. The codebase was then subject to deep analysis and scrutiny, resulting in a series of observations. The problems and their potential solutions are discussed in this document and, whenever possible, we identify common sources for such problems and comment on them as well.

The Fairyproof auditing process follows a routine series of steps:

1. Code Review, Including:

- Project Diagnosis

Understanding the size, scope and functionality of your project's source code based on the specifications, sources, and instructions provided to Fairyproof.

- Manual Code Review

Reading your source code line-by-line to identify potential vulnerabilities.

- Specification Comparison

Determining whether your project's code successfully and efficiently accomplishes or executes its functions according to the specifications, sources, and instructions provided to Fairyproof.

2. Testing and Automated Analysis, Including:

- Test Coverage Analysis

Determining whether the test cases cover your code and how much of your code is exercised or executed when test cases are run.

- Symbolic Execution

Analyzing a program to determine the specific input that causes different parts of a program to execute its functions.

3. Best Practices Review

Reviewing the source code to improve maintainability, security, and control based on the latest established industry and academic practices, recommendations, and research.

— Structure of the document

This report contains a list of issues and comments on all the above source files. Each issue is assigned a severity level based on the potential impact of the issue and recommendations to fix it, if applicable. For ease of navigation, an index by topic and another by severity are both provided at the beginning of the report.

— Documentation

For this audit, we used the following source(s) of truth about how the token issuance function should work:

Website:https://zebra.xyz/

Whitepaper:https://zebra.gitbook.io/zebra-docs

Source Code:

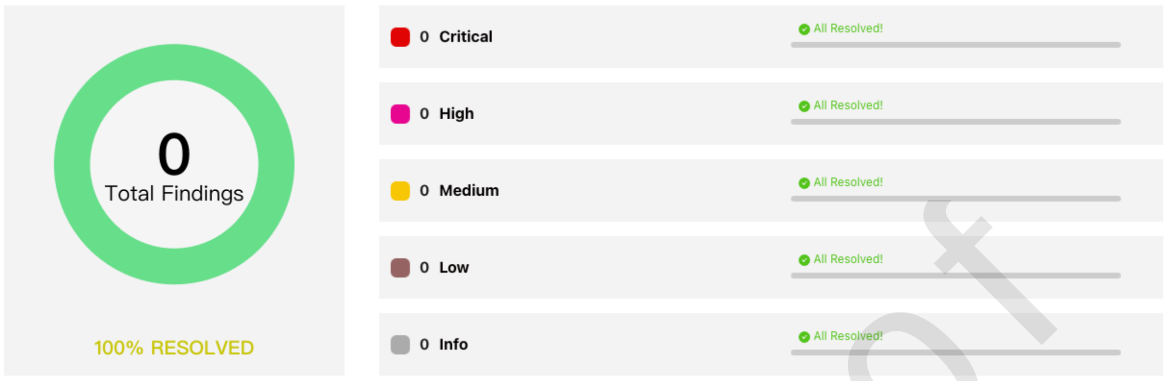
https://github.com/zebra-xyz/v3-contracts

https://github.com/zebra-xyz/universal-router

These were considered the specification, and when discrepancies arose with the actual code behavior, we consulted with the Zebra team or reported an issue.

— Comments from Auditor

Serial Number	Auditor	Audit Time	Result
2023112200012016	Fairyproof Security Team	Nov 21, 2023 - Nov 22, 2023	Passed



Summary:

The Fairyproof security team used its auto analysis tools and manual work to audit the project. During the audit, no issues were uncovered.

02. About Fairyproof

Fairyproof is a leading technology firm in the blockchain industry, providing consulting and security audits for organizations. Fairyproof has developed industry security standards for designing and deploying blockchain applications.

03. Introduction to Zebra

Zebra is a fully permissionless and composable decentralized exchange built on Scroll. It is committed to providing users with a one-stop liquidity service that is cheaper, easier to use, and more secure.

The above description is quoted from relevant documents of Zebra.

04. Major functions of audited code

The audited code mainly implements a DEX similar to Uniswap V3.

05. Coverage of issues

The issues that the Fairyproof team covered when conducting the audit include but are not limited to the following ones:

- Access Control
- Admin Rights
- Arithmetic Precision
- Code Improvement
- Contract Upgrade/Migration
- Delete Trap
- Design Vulnerability
- DoS Attack
- EOA Call Trap
- Fake Deposit
- Function Visibility
- Gas Consumption
- Implementation Vulnerability
- Inappropriate Callback Function
- Injection Attack
- Integer Overflow/Underflow
- IsContract Trap
- Miner's Advantage
- Misc
- Price Manipulation
- Proxy selector clashing
- Pseudo Random Number
- Re-entrancy Attack
- Replay Attack
- Rollback Attack
- Shadow Variable
- Slot Conflict

- Token Issuance
- Tx.origin Authentication
- Uninitialized Storage Pointer

06. Severity level reference

Every issue in this report was assigned a severity level from the following:

Critical severity issues need to be fixed as soon as possible.

High severity issues will probably bring problems and should be fixed.

Medium severity issues could potentially bring problems and should eventually be fixed.

Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

Informational is not an issue or risk but a suggestion for code improvement.

07. Major areas that need attention

Based on the provided source code the Fairyproof team focused on the possible issues and risks related to the following functions or areas.

- Function Implementation

We checked whether or not the functions were correctly implemented.
We didn't find issues or risks in these functions or areas at the time of writing.

- Access Control

We checked each of the functions that could modify a state, especially those functions that could only be accessed by owner or administrator
We didn't find issues or risks in these functions or areas at the time of writing.

- Token Issuance & Transfer

We examined token issuance and transfers for situations that could harm the interests of holders.
We didn't find issues or risks in these functions or areas at the time of writing.

- State Update

We checked some key state variables which should only be set at initialization.
We didn't find issues or risks in these functions or areas at the time of writing.

- Asset Security

We checked whether or not all the functions that transfer assets were safely handled.
We didn't find issues or risks in these functions or areas at the time of writing.

- Miscellaneous

We checked the code for optimization and robustness.
We didn't find issues or risks in these functions or areas at the time of writing.

08. issues by severity

- N/A

09. Issue descriptions

- N/A

10. Recommendations to enhance the overall security

We list some recommendations in this section. They are not mandatory but will enhance the overall security of the system if they are adopted.

- Consider managing the owner's access control with great care and transfering it to a multi-sig wallet or DAO when necessary.

11. Appendices

11.1 Unit Test

1. UniversalRouter.t.sol

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.15;

import 'forge-std/Test.sol';
import {UniversalRouter} from '../../contracts/UniversalRouter.sol';
import {Payments} from '../../contracts/modules/Payments.sol';
import {Constants} from '../../contracts/libraries/Constants.sol';
import {Commands} from '../../contracts/libraries/Commands.sol';
import {MockERC20} from './mock/MockERC20.sol';
import {MockERC1155} from './mock/MockERC1155.sol';
import {Callbacks} from '../../contracts/base/Callbacks.sol';
import {ExampleModule} from '../../contracts/test/ExampleModule.sol';
import {RouterParameters} from '../../contracts/base/RouterImmutables.sol';
import {ERC20} from 'solmate/src/tokens/ERC20.sol';
import 'permit2/src/interfaces/IAllowanceTransfer.sol';

import '@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol';
import '@openzeppelin/contracts/token/ERC1155/IERC1155Receiver.sol';

contract UniversalRouterTest is Test {
    address constant RECIPIENT = address(10);
    uint256 constant AMOUNT = 10 ** 18;

    UniversalRouter router;
    ExampleModule testModule;
    MockERC20 erc20;
    MockERC1155 erc1155;
    Callbacks callbacks;

    function setUp() public {
        RouterParameters memory params = RouterParameters({
            permit2: address(0),
            weth9: address(0),
            steth: address(0),
            wsteth: address(0),
            seaportV1_5: address(0),
            seaportV1_4: address(0),
            openseaConduit: address(0),
            nftxZap: address(0),
            x2y2: address(0),
            foundation: address(0),
            sudoswap: address(0),
            elementMarket: address(0),
            nft20Zap: address(0),
            cryptopunks: address(0),
            looksRareV2: address(0),
            routerRewardsDistributor: address(0),
            looksRareRewardsDistributor: address(0),
            looksRareToken: address(0),
            v2Factory: address(0),
            v3Factory: address(0),
            pairInitCodeHash: bytes32(0),
            poolInitCodeHash: bytes32(0)
        });
        router = new UniversalRouter(params);
        testModule = new ExampleModule();
        erc20 = new MockERC20();
        erc1155 = new MockERC1155();
        callbacks = new Callbacks();
    }

    event ExampleModuleEvent(string message);

    function testCallModule() public {
        uint256 bytecodeSize;
        address theRouter = address(router);
        assembly {
            bytecodeSize := extcodesize(theRouter)
        }
        emit log_uint(bytecodeSize);
    }

    function testSweepToken() public {
        bytes memory commands = abi.encodePacked(bytes1(uint8(Commands.SWEEP)));
        bytes[] memory inputs = new bytes[](1);
        inputs[0] = abi.encode(address(erc20), RECIPIENT, AMOUNT);

        erc20.mint(address(router), AMOUNT);
        assertEq(erc20.balanceOf(RECIPIENT), 0);

        router.execute(commands, inputs);

        assertEq(erc20.balanceOf(RECIPIENT), AMOUNT);
    }

    function testSweepTokenInsufficientOutput() public {
        bytes memory commands = abi.encodePacked(bytes1(uint8(Commands.SWEEP)));
        bytes[] memory inputs = new bytes[](1);
        inputs[0] = abi.encode(address(erc20), RECIPIENT, AMOUNT + 1);

        erc20.mint(address(router), AMOUNT);
        assertEq(erc20.balanceOf(RECIPIENT), 0);

        vm.expectRevert(Payments.InsufficientToken.selector);
        router.execute(commands, inputs);
    }

    function testSweepETH() public {
        bytes memory commands = abi.encodePacked(bytes1(uint8(Commands.SWEEP)));
        bytes[] memory inputs = new bytes[](1);
        inputs[0] = abi.encode(Constants.ETH, RECIPIENT, AMOUNT);

        assertEq(RECIPIENT.balance, 0);

        router.execute{value: AMOUNT}(commands, inputs);

        assertEq(RECIPIENT.balance, AMOUNT);
    }

    function testSweepETHInsufficientOutput() public {
        bytes memory commands = abi.encodePacked(bytes1(uint8(Commands.SWEEP)));
        bytes[] memory inputs = new bytes[](1);
        inputs[0] = abi.encode(Constants.ETH, RECIPIENT, AMOUNT + 1);

        erc20.mint(address(router), AMOUNT);

        vm.expectRevert(Payments.InsufficientETH.selector);
        router.execute(commands, inputs);
    }

    function testSweepERC1155NotFullAmount() public {
        bytes memory commands =
abi.encodePacked(bytes1(uint8(Commands.SWEEP_ERC1155)));
        bytes[] memory inputs = new bytes[](1);
        uint256 id = 0;
        inputs[0] = abi.encode(address(erc1155), RECIPIENT, id, AMOUNT / 2);

        erc1155.mint(address(router), id, AMOUNT);
        assertEq(erc1155.balanceOf(RECIPIENT, id), 0);

        router.execute(commands, inputs);

        assertEq(erc1155.balanceOf(RECIPIENT, id), AMOUNT);
    }

    function testSweepERC1155() public {
        bytes memory commands =
abi.encodePacked(bytes1(uint8(Commands.SWEEP_ERC1155)));
        bytes[] memory inputs = new bytes[](1);
        uint256 id = 0;
        inputs[0] = abi.encode(address(erc1155), RECIPIENT, id, AMOUNT);

        erc1155.mint(address(router), id, AMOUNT);
        assertEq(erc1155.balanceOf(RECIPIENT, id), 0);

        router.execute(commands, inputs);

        assertEq(erc1155.balanceOf(RECIPIENT, id), AMOUNT);
    }

    function testSweepERC1155InsufficientOutput() public {
        bytes memory commands =
abi.encodePacked(bytes1(uint8(Commands.SWEEP_ERC1155)));
        bytes[] memory inputs = new bytes[](1);
        uint256 id = 0;
        inputs[0] = abi.encode(address(erc1155), RECIPIENT, id, AMOUNT + 1);

        erc1155.mint(address(router), id, AMOUNT);

        vm.expectRevert(Payments.InsufficientToken.selector);
        router.execute(commands, inputs);
    }

    function testSupportsInterface() public {
        bool supportsERC1155 =
callbacks.supportsInterface(type(IERC1155Receiver).interfaceId);
        bool supportsERC721 =
callbacks.supportsInterface(type(IERC721Receiver).interfaceId);
        bool supportsERC165 =
callbacks.supportsInterface(type(IERC165).interfaceId);

        assertEq(supportsERC1155, true);
        assertEq(supportsERC721, true);
        assertEq(supportsERC165, true);
    }
}
```

2. UniversalRouter Test Output

```
Running 9 tests for test/foundry-tests/UniversalRouter.t.sol:UniversalRouterTest
[PASS] testCallModule() (gas: 6316)
[PASS] testSupportsInterface() (gas: 7144)
[PASS] testSweepERC1155() (gas: 58012)
[PASS] testSweepERC1155InsufficientOutput() (gas: 49101)
[PASS] testSweepERC1155NotFullAmount() (gas: 57862)
[PASS] testSweepETH() (gas: 52498)
[PASS] testSweepETHInsufficientOutput() (gas: 67986)
[PASS] testSweepToken() (gas: 70721)
[PASS] testSweepTokenInsufficientOutput() (gas: 72125)
Test result: ok. 9 passed; 0 failed; finished in 3.58ms
```

3. ZebraV3Factory Test OutPut

```
ZebraV3Factory
    ✓ owner is deployer (53ms)
    ✓ factory bytecode size
    ✓ pool bytecode size (214ms)
    ✓ initial enabled fee amounts (39ms)
    #createPool
      ✓ succeeds for low fee pool (364ms)
      ✓ succeeds for medium fee pool (132ms)
      ✓ succeeds for high fee pool (146ms)
      ✓ succeeds if tokens are passed in reverse (110ms)
      ✓ fails if token a == token b
      ✓ fails if token a is 0 or token b is 0
      ✓ fails if fee amount is not enabled
      ✓ gas
    #setOwner
      ✓ fails if caller is not owner
      ✓ updates owner
      ✓ emits event
      ✓ cannot be called by original owner
    #enableFeeAmount
      ✓ fails if caller is not owner
      ✓ fails if fee is too great
      ✓ fails if tick spacing is too small
      ✓ fails if tick spacing is too large
      ✓ fails if already initialized
      ✓ sets the fee amount in the mapping
      ✓ emits an event
      ✓ enables pool creation (108ms)


  24 passing (3s)
```

4. ZebraV3Pool Test Output

```
ZebraV3Pool
    ✓ constructor initializes immutables
    ✓ tick transition cannot run twice if zero for one swap ends at fractional
price just below tick (179ms)
    #initialize
      ✓ fails if already initialized
      ✓ fails if starting price is too low
      ✓ fails if starting price is too high
      ✓ can be initialized at MIN_SQRT_RATIO
      ✓ can be initialized at MAX_SQRT_RATIO - 1
      ✓ sets initial variables
      ✓ initializes the first observations slot
      ✓ emits a Initialized event with the input tick
    #increaseObservationCardinalityNext
      ✓ can only be called after initialize
      ✓ emits an event including both old and new
      ✓ does not emit an event for no op call
      ✓ does not change cardinality next if less than current
      ✓ increases cardinality and cardinality next first time
    #mint
      ✓ fails if not initialized
      after initialization
        ✓ protocol fees accumulate as expected during swap (87ms)
        ✓ positions are protected before protocol fee is turned on (91ms)
        ✓ poke is not allowed on uninitialized position (131ms)
        failure cases
          ✓ fails if tickLower greater than tickUpper
          ✓ fails if tickLower less than min tick
          ✓ fails if tickUpper greater than max tick
          ✓ fails if amount exceeds the max (63ms)
          ✓ fails if total amount at tick exceeds the max (113ms)
          ✓ fails if amount is 0
        success cases
          ✓ initial balances
          ✓ initial tick
          above current price
            ✓ transfers token0 only
            ✓ max tick with max leverage
            ✓ works for max tick
            ✓ removing works (48ms)
            ✓ adds liquidity to liquidityGross (104ms)
            ✓ removes liquidity from liquidityGross (73ms)
            ✓ clears tick lower if last position is removed (42ms)
            ✓ clears tick upper if last position is removed (45ms)
            ✓ only clears the tick that is not used at all (70ms)
            ✓ does not write an observation
          including current price
            ✓ price within range: transfers current price of both tokens
            ✓ initializes lower tick
            ✓ initializes upper tick
            ✓ works for min/max tick
            ✓ removing works (57ms)
            ✓ writes an observation (44ms)
          below current price
            ✓ transfers token1 only
            ✓ min tick with max leverage
            ✓ works for min tick
            ✓ removing works (44ms)
            ✓ does not write an observation
      #burn
        ✓ does not clear the position fee growth snapshot if no more liquidity
(99ms)
        ✓ clears the tick if its the last position using it (87ms)
        ✓ clears only the lower tick if upper is still used (115ms)
        ✓ clears only the upper tick if lower is still used (104ms)
      #observe
        ✓ current tick accumulator increases by tick over time
        ✓ current tick accumulator after single swap
        ✓ current tick accumulator after two swaps (84ms)
      miscellaneous mint tests
        ✓ mint to the right of the current price (38ms)
        ✓ mint to the left of the current price (44ms)
        ✓ mint within the current price (42ms)
        ✓ cannot remove more than the entire position (41ms)
        ✓ collect fees within the current price after swap (101ms)
      post-initialize at medium fee
        k (implicit)
          ✓ returns 0 before initialization
          post initialized
            ✓ returns initial liquidity
            ✓ returns in supply in range
            ✓ excludes supply at tick above current tick
            ✓ excludes supply at tick below current tick
            ✓ updates correctly when exiting range (68ms)
            ✓ updates correctly when entering range (62ms)
      limit orders
        ✓ limit selling 0 for 1 at tick 0 thru 1 (133ms)
        ✓ limit selling 1 for 0 at tick 0 thru -1 (79ms)
        fee is on
          ✓ limit selling 0 for 1 at tick 0 thru 1 (83ms)
          ✓ limit selling 1 for 0 at tick 0 thru -1 (82ms)
      #collect
        ✓ works with multiple LPs (130ms)
        works across large increases
          ✓ works just before the cap binds
          ✓ works just after the cap binds
          ✓ works well after the cap binds
        works across overflow boundaries
          ✓ token0 (38ms)
          ✓ token1
          ✓ token0 and token1 (66ms)
      #feeProtocol
        ✓ is initially set to 0
        ✓ can be changed by the owner
        ✓ cannot be changed out of bounds
        ✓ cannot be changed by addresses that are not owner
        ✓ position owner gets full fees when protocol fee is off (41ms)
        ✓ swap fees accumulate as expected (0 for 1) (118ms)
        ✓ swap fees accumulate as expected (1 for 0) (106ms)
        ✓ position owner gets partial fees when protocol fee is on (50ms)
        ✓ fees collected by lp after two swaps should be double one swap (72ms)
        ✓ fees collected after two swaps with fee turned on in middle are fees from
last swap (not confiscatory) (75ms)
        ✓ fees collected by lp after two swaps with intermediate withdrawal (94ms)
        #collectProtocol
          ✓ returns 0 if no fees
          ✓ can collect fees (51ms)
          ✓ fees collected can differ between token0 and token1 (83ms)
      #tickSpacing
        tickSpacing = 12
          post initialize
            ✓ mint can only be called for multiples of 12
            ✓ mint can be called with multiples of 12 (55ms)
            ✓ swapping across gaps works in 1 for 0 direction (184ms)
            ✓ swapping across gaps works in 0 for 1 direction (187ms)
      #flash
        ✓ fails if not initialized (72ms)
        ✓ fails if no liquidity (67ms)
        after liquidity added
          fee off
            ✓ emits an event
            ✓ transfers the amount0 to the recipient
            ✓ transfers the amount1 to the recipient
            ✓ can flash only token0
            ✓ can flash only token1
            ✓ can flash entire token balance
            ✓ no-op if both amounts are 0
            ✓ fails if flash amount is greater than token balance
            ✓ calls the flash callback on the sender with correct fee amounts
            ✓ increases the fee growth by the expected amount
            ✓ fails if original balance not returned in either token (44ms)
            ✓ fails if underpays either token
            ✓ allows donating token0
            ✓ allows donating token1
            ✓ allows donating token0 and token1 together
          fee on
            ✓ emits an event
            ✓ increases the fee growth by the expected amount
            ✓ allows donating token0
            ✓ allows donating token1
            ✓ allows donating token0 and token1 together
      #increaseObservationCardinalityNext
        ✓ cannot be called before initialization
        after initialization
          ✓ oracle starting state after initialization
          ✓ increases observation cardinality next
          ✓ is no op if target is already exceeded
      #setFeeProtocol
        ✓ can only be called by factory owner
        ✓ fails if fee is lt 4 or gt 10
        ✓ succeeds for fee of 4
        ✓ succeeds for fee of 10
        ✓ sets protocol fee
        ✓ can change protocol fee
        ✓ can turn off protocol fee
```

```
    ✓ emits an event when turned on
    ✓ emits an event when turned off
    ✓ emits an event when changed
    ✓ emits an event when unchanged
  #lock
    ✓ cannot reenter from swap callback
  #snapshotCumulativesInside
    ✓ throws if ticks are in reverse order
    ✓ throws if ticks are the same
    ✓ throws if tick lower is too low
    ✓ throws if tick upper is too high
    ✓ throws if tick lower is not initialized
    ✓ throws if tick upper is not initialized
    ✓ is zero immediately after initialize
    ✓ increases by expected amount when time elapses in the range
    ✓ does not account for time increase above range (44ms)
    ✓ does not account for time increase below range (40ms)
    ✓ time increase below range is not counted (69ms)
    ✓ time increase above range is not counted (81ms)
    ✓ positions minted after time spent (101ms)
    ✓ overlapping liquidity is aggregated (80ms)
    ✓ relative behavior of snapshots (71ms)
  fees overflow scenarios
    ✓ up to max uint 128 (68ms)
    ✓ overflow max uint 128 (121ms)
    ✓ overflow max uint 128 after poke burns fees owed to 0 (85ms)
    ✓ two positions at the same snapshot (132ms)
    ✓ two positions 1 wei of fees apart overflows exactly once (163ms)
  swap underpayment tests
    ✓ underpay zero for one and exact in
    ✓ pay in the wrong token zero for one and exact in
    ✓ overpay zero for one and exact in
    ✓ underpay zero for one and exact out (40ms)
    ✓ pay in the wrong token zero for one and exact out
    ✓ overpay zero for one and exact out
    ✓ underpay one for zero and exact in
    ✓ pay in the wrong token one for zero and exact in
    ✓ overpay one for zero and exact in
    ✓ underpay one for zero and exact out
    ✓ pay in the wrong token one for zero and exact out
    ✓ overpay one for zero and exact out


166 passing (19s)
```

5. ZebraV3Router Test Output

```
ZebraV3Pool
    ✓ constructor initializes immutables
    multi-swaps
      ✓ multi-swap (61ms)
2 passing (2s)
```

11.2 External Functions Check Points

1. File: contracts/ZebraV3Pool.sol

contract: ZebraV3Pool is IZebraV3Pool, NoDelegateCall

(Empty fields in the table represent things that are not required or relevant)

Index	Function	StateMutability	Modifier	Param Check	IsUserInterface	Unit Test	Miscellaneous
1	snapshotCumulativesInside(int24,int24)	view	`noDelegateCall`			Passed	
2	observe(uint32[])	view	`noDelegateCall`			Passed	
3	increaseObservationCardinalityNext(uint16)		`lock`, `noDelegateCall`		Yes	Passed	
4	initialize(uint160)				Yes	Passed	
5	mint(address,int24,int24,uint128,bytes)		`lock`		Yes	Passed	
6	collect(address,int24,int24,uint128,uint128)		`lock`		Yes	Passed	
7	burn(int24,int24,uint128)		`lock`		Yes	Passed	
8	swap(address,bool,int256,uint160,bytes)		`noDelegateCall`		Yes	Passed	
9	flash(address,uint256,uint256,bytes)		`lock`, `noDelegateCall`		Yes	Passed	
10	setFeeProtocol(uint8,uint8)		`lock`, `onlyFactoryOwner`			Passed	
11	collectProtocol(address,uint128,uint128)		`lock`, `onlyFactoryOwner`			Passed	

2. File: contracts/SwapRouter.sol

contract: SwapRouter is ISwapRouter, PeripheryImmutableState, PeripheryValidation, PeripheryPaymentsWithFee, Multicall, SelfPermit

(Empty fields in the table represent things that are not required or relevant)

Index	Function	StateMutability	Modifier	Param Check	IsUserInterface	Unit Test	Miscellaneous
1	zebraV3SwapCallback(int256,int256,bytes)				Yes	Passed	
2	exactInputSingle(ExactInputSingleParams)	payable	`checkDeadline(params.deadline)`		Yes	Passed	
3	exactInput(ExactInputParams)	payable	`checkDeadline(params.deadline)`		Yes	Passed	
4	exactOutputSingle(ExactOutputSingleParams)	payable	`checkDeadline(params.deadline)`		Yes	Passed	
5	exactOutput(ExactOutputParams)	payable	`checkDeadline(params.deadline)`		Yes	Passed	

3. File: contracts/ZebraV3Factory.sol

contract: ZebraV3Factory is IZebraV3Factory, ZebraV3PoolDeployer, NoDelegateCall

(Empty fields in the table represent things that are not required or relevant)

Index	Function	StateMutability	Modifier	Param Check	IsUserInterface	Unit Test	Miscellaneous
1	createPool(address,address,uint24)		`noDelegateCall`		Yes	Passed	
2	setOwner(address)					Passed	OnlyOwner
3	enableFeeAmount(uint24,int24)					Passed	OnlyOwner

FAIRYPROOF

https://medium.com/@FairyproofT

https://twitter.com/FairyproofT

https://www.linkedin.com/company/fairyproof-tech

https://t.me/Fairyproof_tech

Reddit: https://www.reddit.com/user/FairyproofTech