Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: Weav3 Finance
Platform: zkSync Network
Language: Solidity
Date: August 14th, 2023

Table of contents

Email: audit@EtherAuthority.io

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by the PodcastPunk team to perform the Security audit of the Weav3 smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on August 14th, 2023.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

- Weav3 is a DEX system that covers multiple contracts, and all contracts have different functions.
    - **Weav3Token:** Contract details minter and mint-related information.
    - **Weav3Factory:** Contract deploys Weav3 pools, manages ownership, control, and protocol fees.
    - **Quoter:** Contract offers quotes for swaps and allows getting the expected amount out or amount in for a given swap without executing the swap.
    - **TickLens:** Contract fetches tick data for pool.
    - **Multicall:** Contract enables multiple methods to be called simultaneously.
    - **NonfungiblePositionManager:** Contract wraps Uniswap V3 positions in ERC721 non-fungible token interface.
    - **NonfungibleTokenPositionDescriptor:** Contract describes NFT token positions and generates data URI for JSON metadata string.
    - **Permit2:** Permit2 handles signature-based transfers in SignatureTransfer and allowance-based transfers in AllowanceTransfer.
    - **DeployUniversalRouter:** A contract is used to set up a permit2 address, weth address, v3factory address.

- The smart contracts have functions like collect, mint, burn, add a Liquidity pool, create a new pool, deploy, deposit, withdraw, execute, receive, dispatch, etc.

Audit scope

Name	Code Review and Security Analysis Report for Weav3 Smart Contracts
Platform	**zkSync Network / Solidity**
File 1	NonfungiblePositionManager.sol
File 1 MD5 Hash	5A0702F887BB6EDCF006FF00BFC27D0D
File 2	NonfungibleTokenPositionDescriptor.sol
File 2 MD5 Hash	40F276FC9F855E0535AB3741E6E9E33C
File 3	Weav3Factory.sol
File 3 MD5 Hash	9272E65F601C395E96AFBBF894B5CA98
File 4	TickLens.sol
File 4 MD5 Hash	DE5D503DCBE84605841212DAD1BCDA7C
File 5	Weav3Token.sol
File 5 MD5 Hash	BA4E8D0D36F210E599C4D00E7775FE51
File 6	Quoter.sol
File 6 MD5 Hash	F622B30828AFE80DE61E543573CCCCA4
File 7	DeployUniversalRouter.sol
File 7 MD5 Hash	FF420C73C12AE904B03BEF2E2426E139
File 8	UniversalRouter.sol
File 8 MD5 Hash	57E35BF94E20124210C307F3C9DA4537
File 9	Multicall.sol
File 9 MD5 Hash	97E35FE1126F33CBFE6E70BA30F14FB8
File 10	Permit2.sol
File 10 MD5 Hash	74251A8577043C101B508D05E2D1E7C3
Audit Date	August 14h, 2023

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 Weav3Token.sol • Name: Weav3Token • Symbol: We3 • Decimals:18 • Maximum Supply: **<u>The owner has control over the following functions:</u>** • Current Minter can set the new minter address. • NFTs are minted based on the amount. • Minter can mint unlimited tokens.	**YES, This is valid.**
File 2 Weav3Factory.sol • Deploys Weav3 pools and manages ownership and control over pool protocol fees. **<u>The owner has control over the following functions:</u>** • Set the enhanced swap fees details. • Current owner can set the new owner address. • Owner can enable the fee amount.	**YES, This is valid.**
File 3 Quote.sol • Allows getting the expected amount out or amount in for a given swap without executing the swap.	**YES, This is valid.**
File 4 TickLens.sol • Contract fetches tick data for pool.	**YES, This is valid.**
File 5 Multicall.sol • Contract enables multiple methods to be called simultaneously.	**YES, This is valid.**
File 6 NonfungiblePositionManager.sol • Name: Weav3 Positions NFT-V1 • Symbol: Weav3-POS	**YES, This is valid.**

<u>Authorization has control over the following functions:</u> ● Authorization can decrease liquidity value. ● Authorization can collect data. ● Authorization can burn a token.	
File 7 NonfungibleTokenPositionDescriptor.sol ● Contract describes NFT token positions and generates data URI for JSON metadata strings.	**YES, This is valid.**
File 8 Permit2.sol ● Permit2 handles signature-based transfers in SignatureTransfer and allowance-based transfers in AllowanceTransfer.	**YES, This is valid.**
File 9 DeployUniversalRouter.sol ● A contract is used to set up a permit2 address, weth address, v3factory address.	**YES, This is valid.**
File 10 UniversalRouter.sol ● A contract is used to execute commands.	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. Also, these contracts contain owner control, which does not make them fully decentralized.

We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 1 medium, 2 low and 5 very low level issues.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Moderated
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Moderated
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Moderated
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Moderated
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: PASSED

Code Quality

This audit scope has 10 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in Weav3 are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the Weav3 Finance.

The PodcastPunk team has not provided unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are well commented on smart contracts.

Documentation

We were given a Weav3 smart contract code in the form of a file. The hash of that code is mentioned above in the table.

As mentioned above, code parts are well commented on. And the logic is straightforward. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official project URL: which provided rich information about the project architecture and tokenomics.

Use of Dependencies

As per our observation, the libraries are used in this smart contracts infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

NonfungiblePositionManager.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	positions	external	Passed	No Issue
3	cachePoolKey	write	Passed	No Issue
4	mint	external	Passed	No Issue
5	isAuthorizedForToken	modifier	Passed	No Issue
6	tokenURI	read	Passed	No Issue
7	baseURI	write	Passed	No Issue
8	increaseLiquidity	external	Passed	No Issue
9	decreaseLiquidity	external	is Authorized For Token	No Issue
10	collect	external	is Authorized For Token	No Issue
11	burn	external	is Authorized For Token	No Issue
12	_getAndIncrementNonce	internal	Passed	No Issue
13	getApproved	read	Passed	No Issue
14	_approve	internal	Passed	No Issue
15	multicall	write	Passed	No Issue
16	DOMAIN_SEPARATOR	read	Passed	No Issue
17	permit	external	Passed	No Issue
18	createAndInitializePoolIfNecessary	external	Passed	No Issue
19	weav3MintCallback	external	Passed	No Issue
20	addLiquidity	internal	Passed	No Issue
21	checkDeadline	modifier	Passed	No Issue
22	selfPermit	write	Passed	No Issue
23	selfPermitIfNecessary	external	Passed	No Issue
24	selfPermitAllowed	write	Passed	No Issue
25	selfPermitAllowedIfNecessary	external	Passed	No Issue

NonfungibleTokenPositionDescriptor.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	nativeCurrencyLabel	read	Passed	No Issue
3	tokenURI	external	Passed	No Issue
4	flipRatio	read	Passed	No Issue
5	tokenRatioPriority	read	Passed	No Issue

Weav3Factory.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	createPool	external	Missing required error message	Refer Audit Findings
3	setEnhancedSwapFee	write	Function input parameters lack of check, Centralization Risk,Missing required error message	Refer Audit Findings
4	setOwner	external	Missing required error message, Function input parameters lack of check	Refer Audit Findings
5	enableFeeAmount	write	Missing required error message	Refer Audit Findings
6	deploy	internal	Passed	No Issue
7	checkNotDelegateCall	read	Passed	No Issue
8	noDelegateCall	modifier	Passed	No Issue

Weav3Token.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	setMinter	external	Function input parameters lack of check	Refer Audit Findings
3	initialMint	external	Function input parameters lack of check	Refer Audit Findings
4	approve	external	Passed	No Issue
5	_mint	internal	Passed	No Issue
6	_transfer	internal	Passed	No Issue
7	transfer	external	Passed	No Issue
8	transferFrom	external	Passed	No Issue
9	mint	external	Unlimited Minting	Refer Audit Findings

WETH.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	deposit	write	Passed	No Issue
3	withdraw	write	Passed	No Issue
4	totalSupply	read	Passed	No Issue
5	approve	write	Passed	No Issue

6	transfer	write	Passed	No Issue
7	transferFrom	write	Passed	No Issue

Quoter.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getPool	read	Passed	No Issue
3	weav3SwapCallback	external	Unused function parameter	Refer Audit Findings
4	parseRevertReason	write	Passed	No Issue
5	quoteExactInputSingle	write	Passed	No Issue
6	quoteExactInput	external	Passed	No Issue
7	quoteExactOutputSingle	write	Passed	No Issue
8	quoteExactOutput	external	Passed	No Issue

TickLens.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getPopulatedTicksInWord	read	Passed	No Issue

Multicall.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	multicall	write	Infinite loops possibility	Refer Audit Findings

DeployUniversalRouter.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	setUp	write	Passed	No Issue
3	setUp	write	Passed	No Issue
4	run	external	Passed	No Issue
5	mapUnsupported	internal	Passed	No Issue

UniversalRouter.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	checkDeadline	modifier	Passed	No Issue
3	execute	external	check Deadline	No Issue
4	execute	write	Passed	No Issue
5	successRequired	internal	Passed	No Issue
6	receive	external	Passed	No Issue
7	dispatch	internal	Passed	No Issue
8	execute	external	Passed	No Issue
9	callAndTransfer721	internal	Passed	No Issue
10	callAndTransfer1155	internal	Passed	No Issue
11	getValueAndData	internal	Passed	No Issue
12	collectRewards	external	Passed	No Issue

Permit2.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	approve	external	Passed	No Issue
3	permit	external	Passed	No Issue
4	permit	external	Passed	No Issue
5	transferFrom	external	Passed	No Issue
6	transferFrom	external	Passed	No Issue
7	_transfer	write	Passed	No Issue
8	lockdown	external	Passed	No Issue
9	invalidateNonces	external	Passed	No Issue
10	_updateApproval	write	Passed	No Issue
11	permitTransferFrom	external	Passed	No Issue
12	permitWitnessTransferFrom	external	Passed	No Issue
13	_permitTransferFrom	write	Passed	No Issue
14	permitTransferFrom	external	Passed	No Issue
15	permitWitnessTransferFrom	external	Passed	No Issue
16	_permitTransferFrom	write	Passed	No Issue
17	invalidateUnorderedNonces	external	Passed	No Issue
18	bitmapPositions	write	Passed	No Issue
19	_useUnorderedNonce	internal	Passed	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No critical severity vulnerabilities were found in the contract code.

High Severity

No high severity vulnerabilities were found in the contract code.

Medium

(1) Centralization Risk: **Weav3Factory.sol**

Function: setEnhancedSwapFee()

```
function setEnhancedSwapFee(bool enable, uint24 _enhancedSwapfee, address receiver) public {
    require(msg.sender == owner);
    enhancedSwapState = enable;
    enhancedSwapFee = _enhancedSwapfee;
    receiverEnhancedSwapFee = receiver;
}
```

In the function setEnhancedSwapFee(), the owner can set 100% fees. They are not validated; it should not be set 100% fee out of 100%. There should be limited fees.

Resolution: It is advised to set a max fee limit for enhanced Swap fees.

Low

(1) Infinite loops possibility: **Multicall.sol**

As array elements will increase, then it will cost more and more gas. And eventually, it will stop all the functionality. After several hundreds of transactions, all those functions depending on it will stop. We suggest avoiding loops. For example, use mapping to store the array index. And query that data directly, instead of looping through all the elements to find an element.

Resolution: Adjust logic to replace loops with mapping or other code structure.

- multicall() - data.length.

(2) Function input parameters lack of check:

Some functions require validation before execution.

Functions are:

Weav3Token.sol

- initialMint
- setMinter

Weav3Factory.sol

- setEnhancedSwapFee() - its owner function.
- setOwner = _owner

Resolution: We suggest using validation like for numerical variables that should be greater than 0 and for address type check variables that are not address(0). For percentage type variables, values should have some range like minimum 0 and maximum 100.

Very Low / Informational / Best practices:

(1) Missing required error message:

Weav3Factory.sol

Function: setEnhancedSwapFee(), setOwner()

```
function setEnhancedSwapFee(bool enable, uint24 _enhancedSwapfee, address receiver) public {
    require(msg.sender == owner);
    enhancedSwapState = enable;
    enhancedSwapFee = _enhancedSwapfee;
    receiverEnhancedSwapFee = receiver;
}

/// @inheritdoc IWeav3Factory
function setOwner(address _owner) external override {
    require(msg.sender == owner);
    emit OwnerChanged(owner, _owner);
    owner = _owner;
}
```

Function: enableFeeAmount()

```
/// @inheritdoc IWeav3Factory
function enableFeeAmount(uint24 fee, int24 tickSpacing) public override {
    require(msg.sender == owner);
    require(fee < 1000000);
    // tick spacing is capped at 16384 to prevent the situation where tickSpacing is so large that
    // TickBitmap#nextInitializedTickWithinOneWord overflows int24 container from a valid tick
    // 16384 ticks represents a >5x price change with ticks of 1 bips
    require(tickSpacing > 0 && tickSpacing < 16384);
    require(feeAmountTickSpacing[fee] == 0);

    feeAmountTickSpacing[fee] = tickSpacing;
    emit FeeAmountEnabled(fee, tickSpacing);
}
```

Function: createPool()

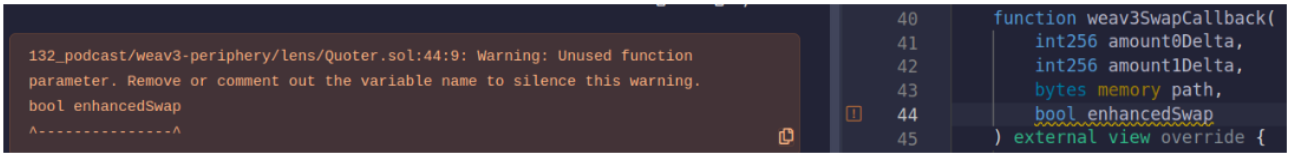
```
/// @inheritdoc IWeav3Factory
function createPool(
    address tokenA,
    address tokenB,
    uint24 fee
) external override noDelegateCall returns (address pool) {
    require(tokenA != tokenB);
    (address token0, address token1) = tokenA < tokenB ? (tokenA, tokenB) : (tokenB, tokenA);
    require(token0 != address(0));
    int24 tickSpacing = feeAmountTickSpacing[fee];
    require(tickSpacing != 0);
    require(getPool[token0][token1][fee] == address(0));
    pool = deploy(address(this), token0, token1, fee, tickSpacing);
    getPool[token0][token1][fee] = pool;
    // populate mapping in the reverse direction, deliberate choice to avoid the cost of comparing addresses
    getPool[token1][token0][fee] = pool;
    emit PoolCreated(token0, token1, fee, tickSpacing, pool);
}
```

There is no set error message in the required statement.

Resolution: We suggest setting relevant error messages to identify the failure of the transaction.

(2) Unused function parameter: **Quoter.sol**

bool enhancedSwap: warning



Warning: Unused function parameter. Remove or comment out the variable name to silence this warning.

Resolution: We suggest removing or commenting on unused function parameters.

(3) Function state mutability: **Quoter.sol**

Warning from PoolAddress library:

```
132_podcast/weav3-periphery/libraries/PoolAddress.sol:29:5: Warning: Function state
mutability can be restricted to pure
function getInitHashCode() public view returns (bytes32) {
^ (Relevant source part starts here and spans across multiple lines).
```

Function state mutability can be restricted to pure.
function getInitHashCode() public view returns (bytes32).

Resolution: We suggest removing the "view" keyword.

(4) Hardcoded token addresses: **NonfungibleTokenPositionDescriptor.sol**

Contract: NonfungibleTokenPositionDescriptor

```
contract NonfungibleTokenPositionDescriptor is INonfungibleTokenPositionDescriptor {
    address private constant USDC = 0x0faF6df7054946141266420b43783387A78d82A9;
    address private constant USDT = 0xdAC17F958D2ee523a2206206994597C13D831ec7;
    address private constant DAI = 0x6B175474E89094C44Da98b954EedeAC495271d0F;
    address private constant TBTC = 0x8dAEBADE922dF735c38C80C7eBD708Af50815fAa;
    address private constant WBTC = 0x2260FAC5E5542a773Aa44fBCfeDf7C193bc2C599;
```

In the buy function, for USDT and WBTC assets ERC20 tokens are hardcoded.

Resolution: We suggest always making sure hardcoded addresses are correct token addresses before deploying contracts.

(5) Unlimited Minting: **NonfungibleTokenPositionDescriptor.sol**

Function: mint()

```
function mint(address account, uint256 amount) external returns (bool) {    infinite gas
    require(msg.sender == minter);
    _mint(account, amount);
    return true;
}
```

Minter can mint unlimited tokens.

Resolution: We suggest putting a minting limit.

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

NonfungiblePositionManager.sol

- decreaseLiquidity: Authorization can decrease liquidity value.
- collect: Authorization can collect data.
- burn: Authorization can burn a token.

Weav3Factory.sol

- setEnhancedSwapFee: Owner can set enhanced swap fees.
- setOwner: Current owner can set new owner address.
- enableFeeAmount: Owner can set enable fee amount.

Weav3Token.sol

- setMinter: Minter address can be set by the owner.
- initialMint: initial Mint address can be set by the owner.
- mint: Minter can mint tokens.

To make the smart contract 100% decentralized, we suggest renouncing ownership in the smart contract once its function is completed.

Conclusion

We were given a contract code in the form of a file. And we have used all possible tests based on given objects as files. We had observed 1 medium, 2 low and 5 Informational severity issues in the smart contracts. but those are not critical. **So, the smart contracts are ready for the mainnet deployment.**

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

The security state of the reviewed contract, based on standard audit procedure scope, is **"Secured".**

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

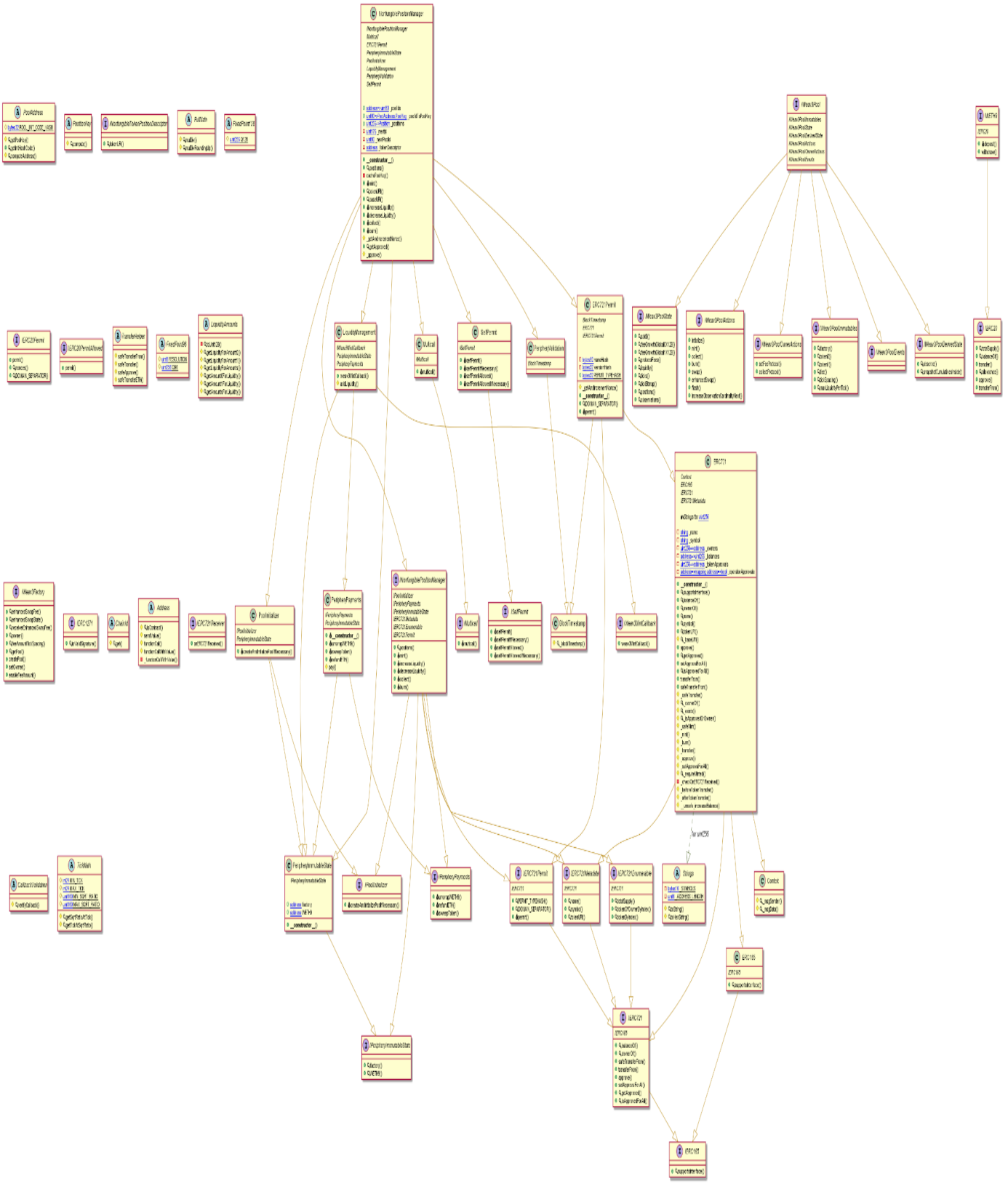
Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.

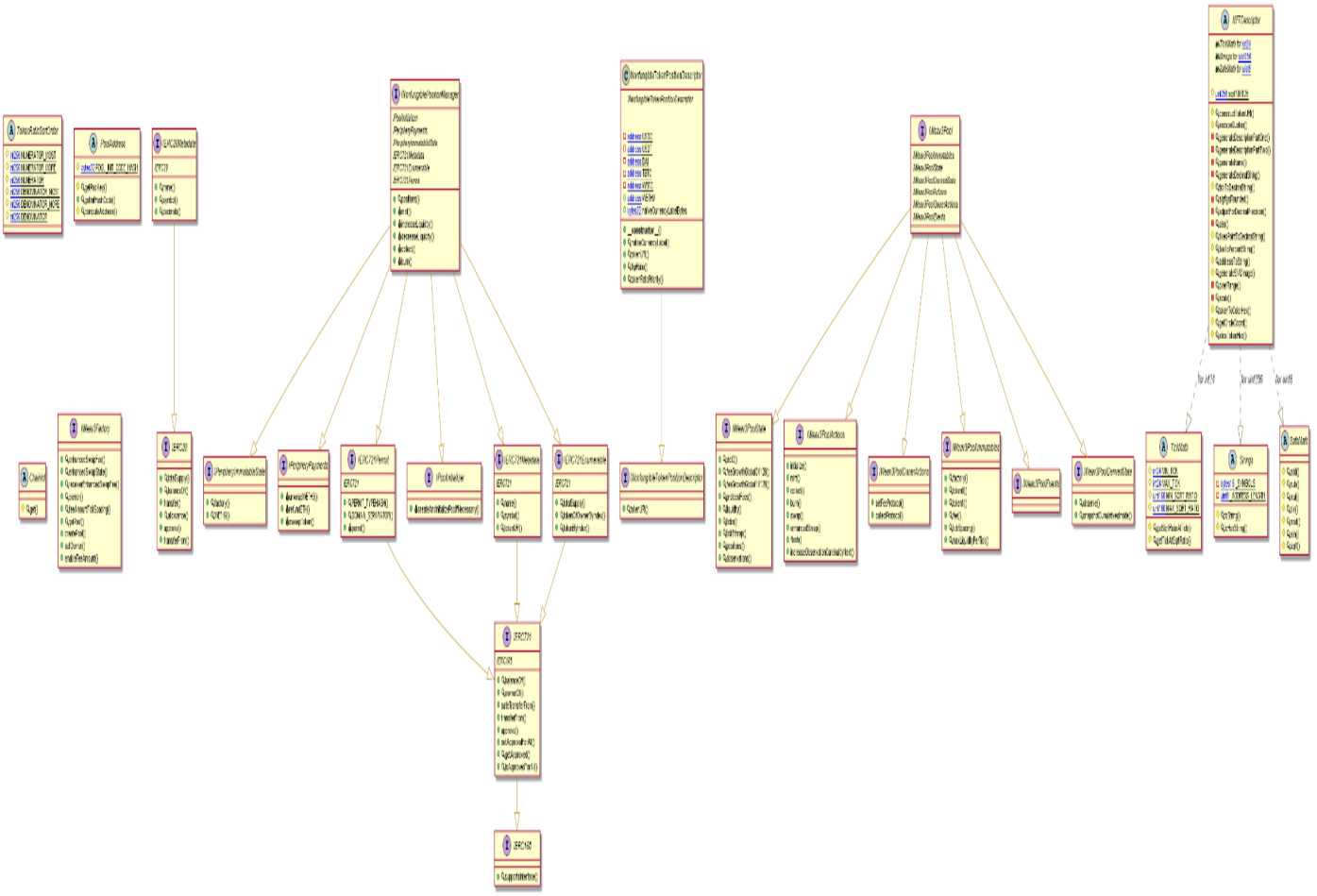
Appendix

Code Flow Diagram - Weav3 Finance

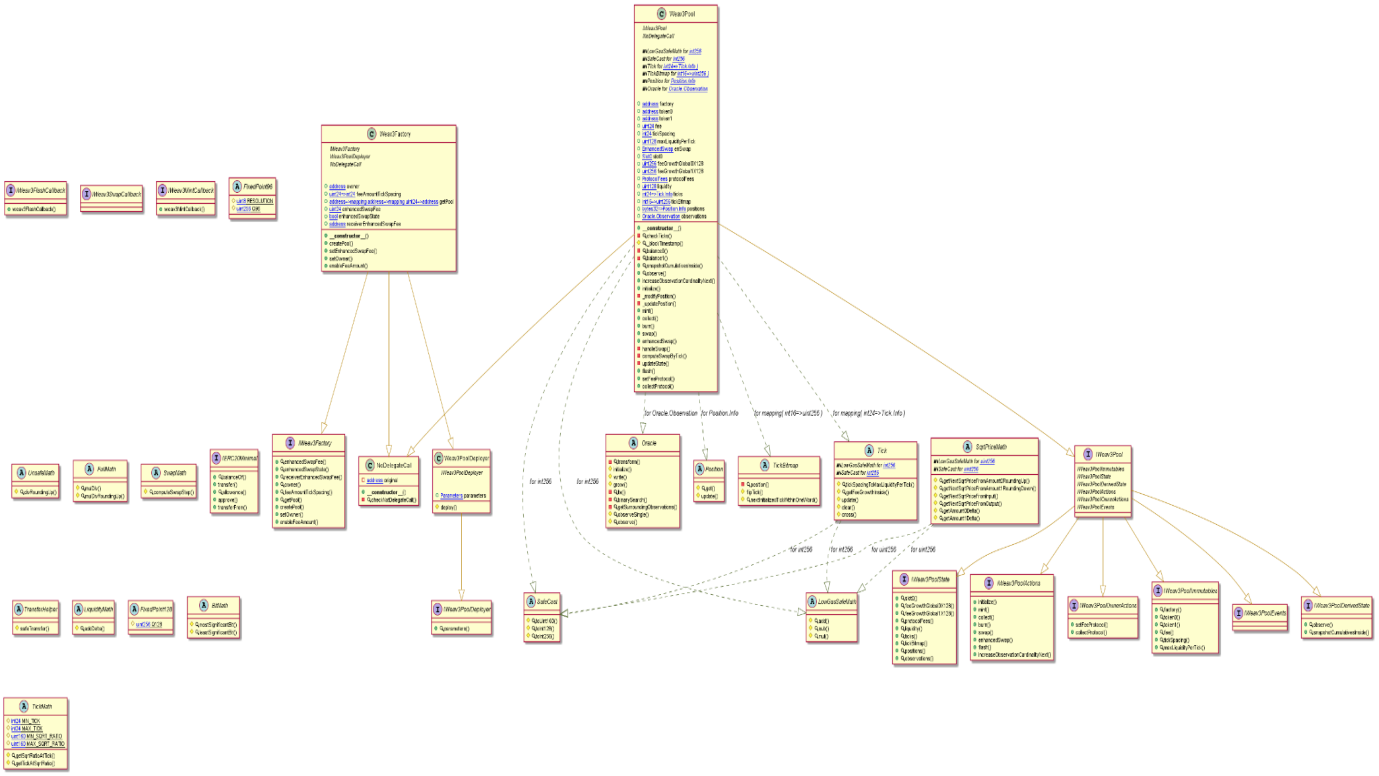
NonfungiblePositionManager Diagram



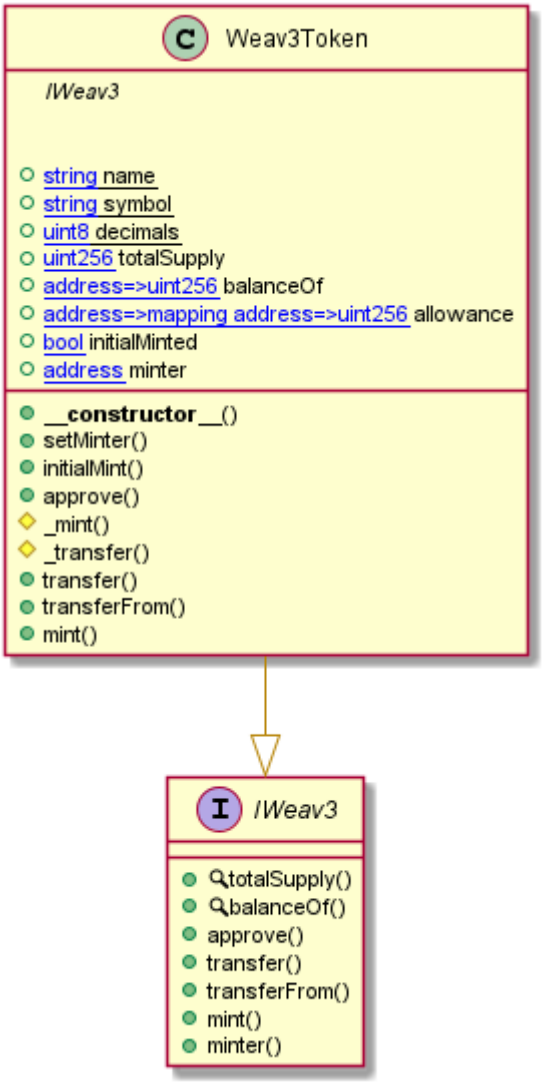
NonfungibleTokenPositionDescriptor Diagram



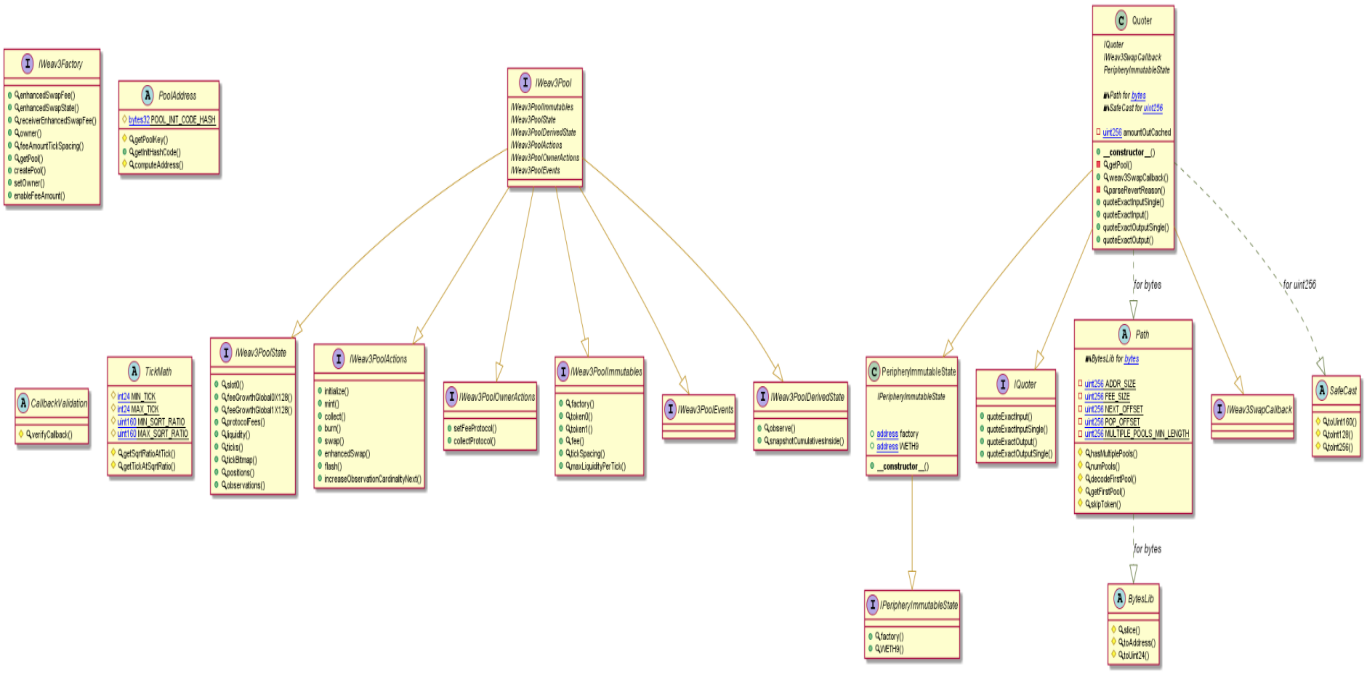
Weav3Factory Diagram



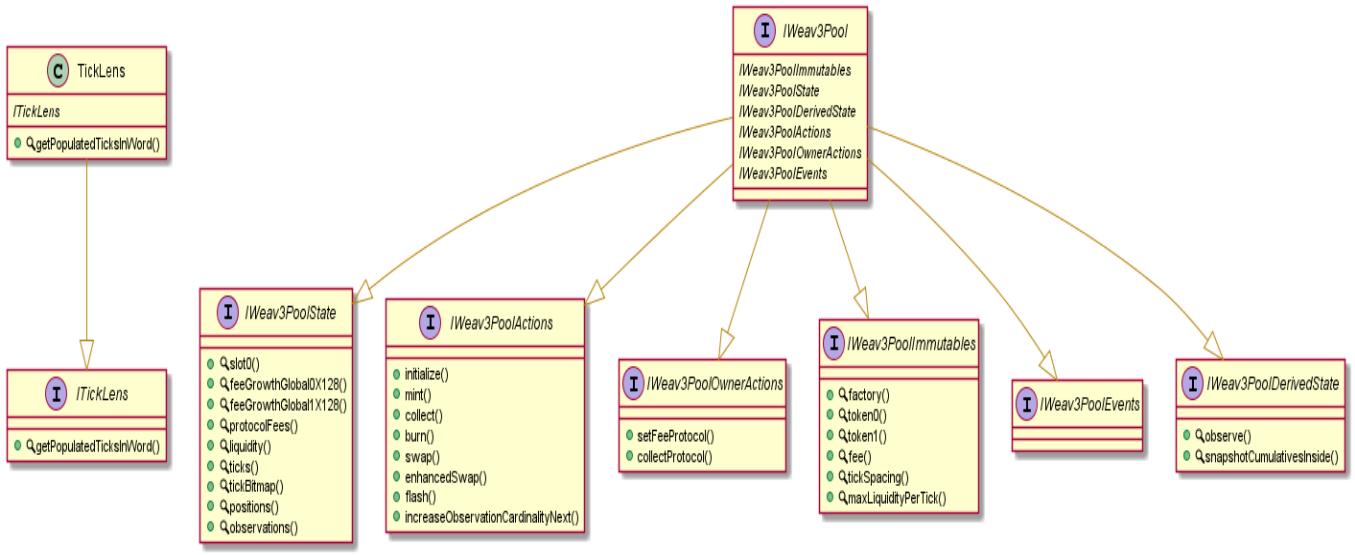
Weav3Token Diagram



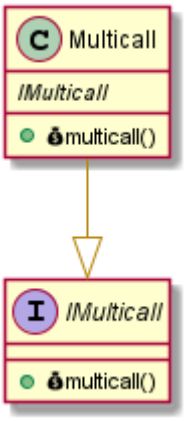
Quoter Diagram



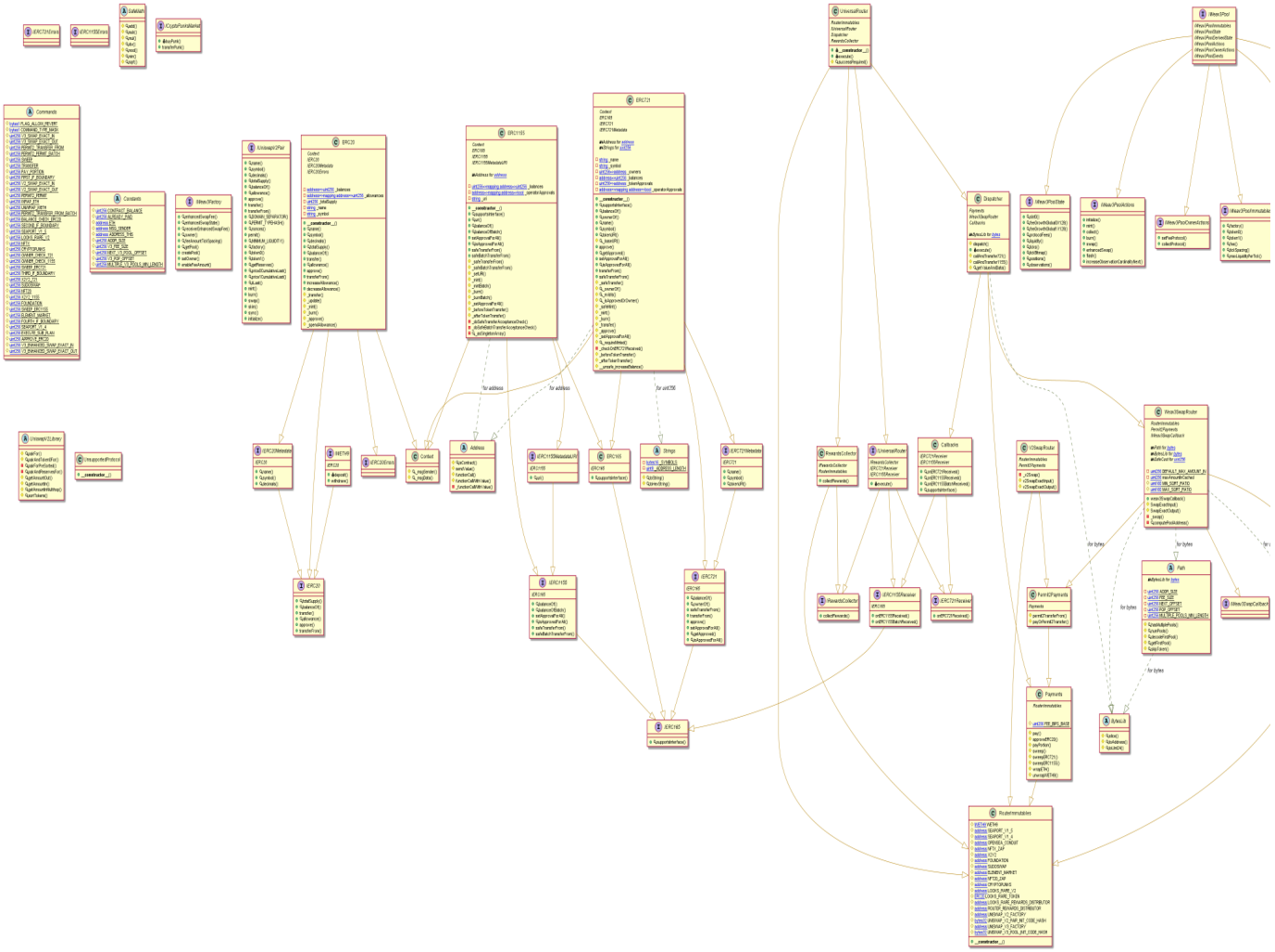
TickLens Diagram



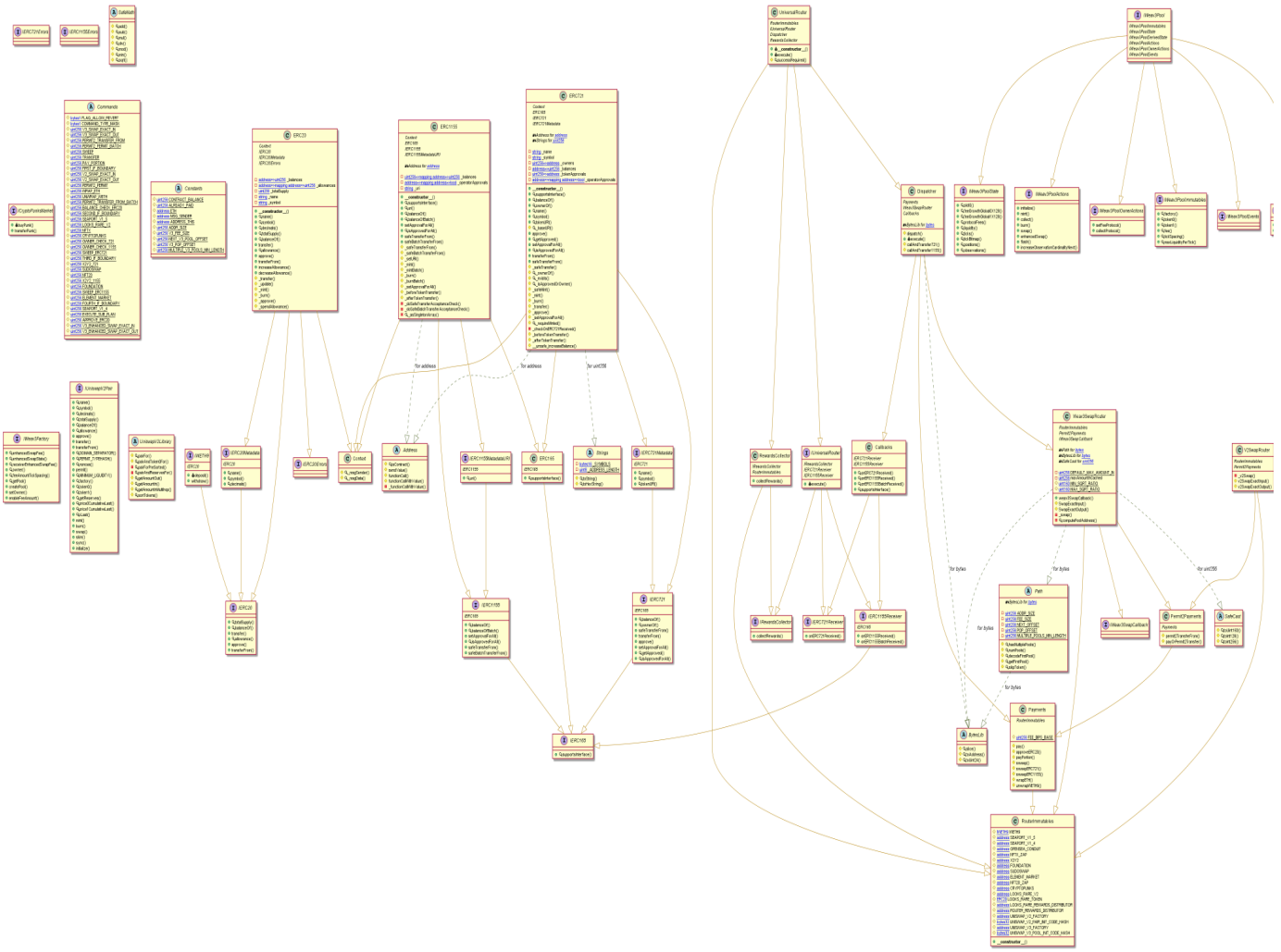
Multicall Diagram



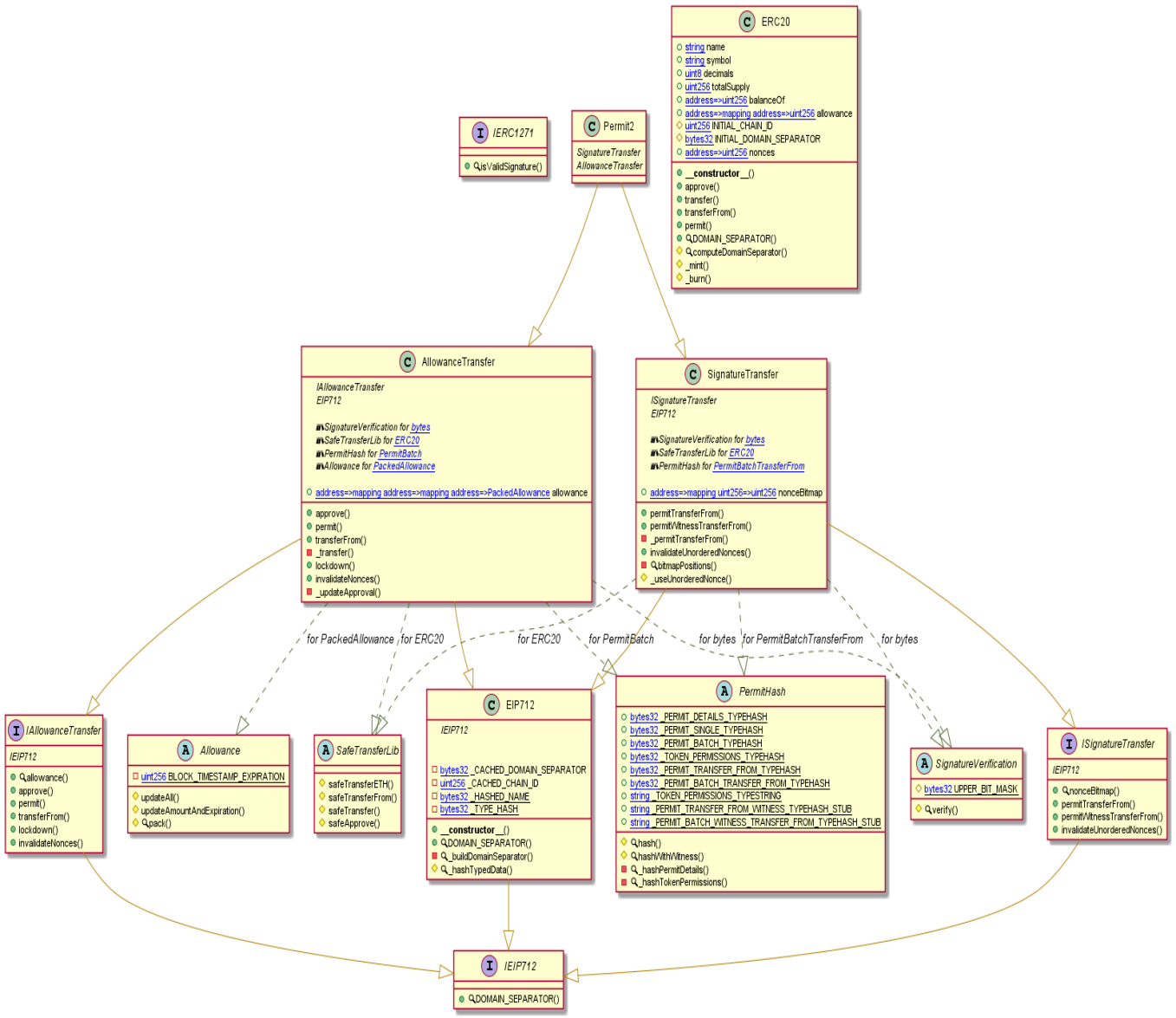
DeployUniversalRouter Diagram



UniversalRouter Diagram



Permit2 Diagram



Slither Results Log

Slither is a Solidity static analysis framework that uses vulnerability detectors, displays contract details, and provides an API for writing custom analyses. It helps developers identify vulnerabilities, improve code comprehension, and prototype custom analyses quickly. The analysis includes a report with warnings and errors, allowing developers to quickly prototype and fix issues.

We did the analysis of the project altogether. Below are the results.

Slither log >> NonfungiblePositionManager.sol

```
Multicall.multicall(bytes[]) (NonfungiblePositionManager.sol#546-562) has external calls inside a loop: (success,result) = add
ress(this).delegatecall(data[i]) (NonfungiblePositionManager.sol#549)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (NonfungiblePositionManager.sol#1610)' in ERC721
._checkOnERC721Received(address,address,uint256,bytes) (NonfungiblePositionManager.sol#1603-1625) potentially used before decl
aration: retval == IERC721Receiver.onERC721Received.selector (NonfungiblePositionManager.sol#1611)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (NonfungiblePositionManager.sol#1612)' in ERC721
._checkOnERC721Received(address,address,uint256,bytes) (NonfungiblePositionManager.sol#1603-1625) potentially used before decl
aration: reason.length == 0 (NonfungiblePositionManager.sol#1613)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (NonfungiblePositionManager.sol#1612)' in ERC721
._checkOnERC721Received(address,address,uint256,bytes) (NonfungiblePositionManager.sol#1603-1625) potentially used before decl
aration: revert(uint256,uint256)(32 + reason,mload(uint256)(reason)) (NonfungiblePositionManager.sol#1618)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in NonfungiblePositionManager.mint(INonfungiblePositionManager.MintParams) (NonfungiblePositionManager.sol#2144-219
8):
        External calls:
        - (liquidity,amount0,amount1,pool) = addLiquidity(AddLiquidityParams(params.token0,params.token1,params.fee,address(th
is),params.tickLower,params.tickUpper,params.amount0Desired,params.amount1Desired,params.amount0Min,params.amount1Min),factory
) (NonfungiblePositionManager.sol#2156-2170)
                - (amount0,amount1) = pool.mint(params.recipient,params.tickLower,params.tickUpper,liquidity,abi.encode(MintCa
llbackData(poolKey,msg.sender))) (NonfungiblePositionManager.sol#2025-2031)
        State variables written after the call(s):
        - _mint(params.recipient,(tokenId = _nextId ++)) (NonfungiblePositionManager.sol#2172)
                - _balances[to] += 1 (NonfungiblePositionManager.sol#1470)
        - _mint(params.recipient,(tokenId = _nextId ++)) (NonfungiblePositionManager.sol#2172)
        - poolId = cachePoolKey(address(pool),PoolAddress.PoolKey(params.token0,params.token1,params.fee)) (NonfungiblePositio
nManager.sol#2178-2182)
                - _poolIds[pool] = (poolId = _nextPoolId ++) (NonfungiblePositionManager.sol#2138)
        - _mint(params.recipient,(tokenId = _nextId ++)) (NonfungiblePositionManager.sol#2172)
                - _owners[tokenId] = to (NonfungiblePositionManager.sol#1473)
```

```
        - poolId = cachePoolKey(address(pool),PoolAddress.PoolKey(params.token0,params.token1,params.fee)) (NonfungiblePositio
nManager.sol#2178-2182)
                - _poolIdToPoolKey[poolId] = poolKey (NonfungiblePositionManager.sol#2139)
        - poolId = cachePoolKey(address(pool),PoolAddress.PoolKey(params.token0,params.token1,params.fee)) (NonfungiblePositio
nManager.sol#2178-2182)
                - _poolIds[pool] = (poolId = _nextPoolId ++) (NonfungiblePositionManager.sol#2138)
        - _positions[tokenId] = Position(0,address(0),poolId,params.tickLower,params.tickUpper,liquidity,feeGrowthInside0LastX
128,feeGrowthInside1LastX128,0,0) (NonfungiblePositionManager.sol#2184-2195)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in NonfungiblePositionManager.collect(INonfungiblePositionManager.CollectParams) (NonfungiblePositionManager.sol#23
27-2393):
        External calls:
        - pool.burn(position.tickLower,position.tickUpper,0) (NonfungiblePositionManager.sol#2349)
        - (amount0,amount1) = pool.collect(recipient,position.tickLower,position.tickUpper,amount0Collect,amount1Collect) (Non
fungiblePositionManager.sol#2380-2386)
        Event emitted after the call(s):
        - Collect(params.tokenId,recipient,amount0Collect,amount1Collect) (NonfungiblePositionManager.sol#2392)
Reentrancy in NonfungiblePositionManager.decreaseLiquidity(INonfungiblePositionManager.DecreaseLiquidityParams) (NonfungiblePo
sitionManager.sol#2274-2324):
        External calls:
        - (amount0,amount1) = pool.burn(position.tickLower,position.tickUpper,params.liquidity) (NonfungiblePositionManager.so
l#2291)
        Event emitted after the call(s):
        - DecreaseLiquidity(params.tokenId,params.liquidity,amount0,amount1) (NonfungiblePositionManager.sol#2323)
Reentrancy in NonfungiblePositionManager.increaseLiquidity(INonfungiblePositionManager.IncreaseLiquidityParams) (NonfungiblePo
sitionManager.sol#2214-2271):
```

```
ERC721Permit.permit(address,uint256,uint256,uint8,bytes32,bytes32) (NonfungiblePositionManager.sol#1715-1745) uses timestamp f
or comparisons
        Dangerous comparisons:
        - require(bool,string)(_blockTimestamp() <= deadline,Permit expired) (NonfungiblePositionManager.sol#1723)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

FullMath.mulDiv(uint256,uint256,uint256) (NonfungiblePositionManager.sol#257-290) uses assembly
        - INLINE ASM (NonfungiblePositionManager.sol#264-268)
        - INLINE ASM (NonfungiblePositionManager.sol#272-274)
        - INLINE ASM (NonfungiblePositionManager.sol#282-284)
        - INLINE ASM (NonfungiblePositionManager.sol#285-288)
Multicall.multicall(bytes[]) (NonfungiblePositionManager.sol#546-562) uses assembly
        - INLINE ASM (NonfungiblePositionManager.sol#554-556)
ChainId.get() (NonfungiblePositionManager.sol#1055-1059) uses assembly
        - INLINE ASM (NonfungiblePositionManager.sol#1056-1058)
Address.isContract(address) (NonfungiblePositionManager.sol#1063-1070) uses assembly
        - INLINE ASM (NonfungiblePositionManager.sol#1066-1068)
Address._functionCallWithValue(address,bytes,uint256,string) (NonfungiblePositionManager.sol#1109-1131) uses assembly
        - INLINE ASM (NonfungiblePositionManager.sol#1123-1126)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (NonfungiblePositionManager.sol#1603-1625) uses assembly
        - INLINE ASM (NonfungiblePositionManager.sol#1617-1619)
TickMath.getTickAtSqrtRatio(uint160) (NonfungiblePositionManager.sol#1808-1950) uses assembly
        - INLINE ASM (NonfungiblePositionManager.sol#1815-1819)
        - INLINE ASM (NonfungiblePositionManager.sol#1820-1824)
        - INLINE ASM (NonfungiblePositionManager.sol#1825-1829)
        - INLINE ASM (NonfungiblePositionManager.sol#1830-1834)
        - INLINE ASM (NonfungiblePositionManager.sol#1835-1839)
        - INLINE ASM (NonfungiblePositionManager.sol#1840-1844)
        - INLINE ASM (NonfungiblePositionManager.sol#1845-1849)
        - INLINE ASM (NonfungiblePositionManager.sol#1850-1853)
        - INLINE ASM (NonfungiblePositionManager.sol#1860-1865)
        - INLINE ASM (NonfungiblePositionManager.sol#1866-1871)
        - INLINE ASM (NonfungiblePositionManager.sol#1872-1877)
        - INLINE ASM (NonfungiblePositionManager.sol#1878-1883)
        - INLINE ASM (NonfungiblePositionManager.sol#1884-1889)
        - INLINE ASM (NonfungiblePositionManager.sol#1890-1895)
```

```
        - INLINE ASM (NonfungiblePositionManager.sol#1878-1883)
        - INLINE ASM (NonfungiblePositionManager.sol#1884-1889)
        - INLINE ASM (NonfungiblePositionManager.sol#1890-1895)
        - INLINE ASM (NonfungiblePositionManager.sol#1896-1901)
        - INLINE ASM (NonfungiblePositionManager.sol#1902-1907)
        - INLINE ASM (NonfungiblePositionManager.sol#1908-1913)
        - INLINE ASM (NonfungiblePositionManager.sol#1914-1919)
        - INLINE ASM (NonfungiblePositionManager.sol#1920-1925)
        - INLINE ASM (NonfungiblePositionManager.sol#1926-1931)
        - INLINE ASM (NonfungiblePositionManager.sol#1932-1937)
        - INLINE ASM (NonfungiblePositionManager.sol#1938-1942)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Pragma version=0.8.0 (NonfungiblePositionManager.sol#2) allows old versions
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Multicall.multicall(bytes[]) (NonfungiblePositionManager.sol#546-562):
        - (success,result) = address(this).delegatecall(data[i]) (NonfungiblePositionManager.sol#549)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (NonfungiblePositionManager.sol#789-798):
        - (success,data) = token.call(abi.encodeWithSelector(IERC20.transferFrom.selector,from,to,value)) (NonfungiblePosition
Manager.sol#795-796)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (NonfungiblePositionManager.sol#800-807):
        - (success,data) = token.call(abi.encodeWithSelector(IERC20.transfer.selector,to,value)) (NonfungiblePositionManager.s
ol#805)
Low level call in TransferHelper.safeApprove(address,address,uint256) (NonfungiblePositionManager.sol#809-816):
        - (success,data) = token.call(abi.encodeWithSelector(IERC20.approve.selector,to,value)) (NonfungiblePositionManager.so
l#814)
Low level call in TransferHelper.safeTransferETH(address,uint256) (NonfungiblePositionManager.sol#818-821):
        - (success) = to.call{value: value}(new bytes(0)) (NonfungiblePositionManager.sol#819)
Low level call in Address.sendValue(address,uint256) (NonfungiblePositionManager.sol#1072-1077):
        - (success) = recipient.call{value: amount}() (NonfungiblePositionManager.sol#1075)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (NonfungiblePositionManager.sol#1109-1131):
        - (success,returndata) = target.call{value: weiValue}(data) (NonfungiblePositionManager.sol#1117)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Function IPeripheryImmutableState.WETH9() (NonfungiblePositionManager.sol#49) is not in mixedCase
Function IERC721Permit.PERMIT_TYPEHASH() (NonfungiblePositionManager.sol#110) is not in mixedCase
Function IERC721Permit.DOMAIN_SEPARATOR() (NonfungiblePositionManager.sol#112) is not in mixedCase
Function IERC20Permit.DOMAIN_SEPARATOR() (NonfungiblePositionManager.sol#579) is not in mixedCase
Variable PeripheryImmutableState.WETH9 (NonfungiblePositionManager.sol#779) is not in mixedCase
Function ERC721.__unsafe_increaseBalance(address,uint256) (NonfungiblePositionManager.sol#1667-1669) is not in mixedCase
Function ERC721Permit.DOMAIN_SEPARATOR() (NonfungiblePositionManager.sol#1695-1707) is not in mixedCase
Parameter LiquidityManagement.addLiquidity(LiquidityManagement.AddLiquidityParams,address)._factory (NonfungiblePositionManage
r.sol#1996) is not in mixedCase
Variable NonfungiblePositionManager._poolIds (NonfungiblePositionManager.sol#2071) is not in mixedCase
Variable NonfungiblePositionManager._poolIdToPoolKey (NonfungiblePositionManager.sol#2074) is not in mixedCase
Variable NonfungiblePositionManager._positions (NonfungiblePositionManager.sol#2077) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (NonfungiblePositionManager.sol#1180)" inContext (NonfungiblePositionManager.sol#1174-1183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
```

```
FixedPoint128.slitherConstructorConstantVariables() (NonfungiblePositionManager.sol#306-308) uses literals with too many digit
s:
        - Q128 = 0x100000000000000000000000000000000 (NonfungiblePositionManager.sol#307)
FixedPoint96.slitherConstructorConstantVariables() (NonfungiblePositionManager.sol#890-893) uses literals with too many digits
:
        - Q96 = 0x1000000000000000000000000 (NonfungiblePositionManager.sol#892)
TickMath.getSqrtRatioAtTick(int24) (NonfungiblePositionManager.sol#1778-1806) uses literals with too many digits:
        - ratio = 0x100000000000000000000000000000000 (NonfungiblePositionManager.sol#1782)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

NonfungiblePositionManager (NonfungiblePositionManager.sol#2039-2419) does not implement functions:
        - IERC721Enumerable.tokenByIndex(uint256) (NonfungiblePositionManager.sol#148)
        - IERC721Enumerable.tokenOfOwnerByIndex(address,uint256) (NonfungiblePositionManager.sol#146)
        - IERC721Enumerable.totalSupply() (NonfungiblePositionManager.sol#144)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

PoolAddress.POOL_INIT_CODE_HASH (NonfungiblePositionManager.sol#6) is never used in PoolAddress (NonfungiblePositionManager.so
l#5-31)
TickMath.MAX_TICK (NonfungiblePositionManager.sol#1773) is never used in TickMath (NonfungiblePositionManager.sol#1771-1951)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
NonfungiblePositionManager.sol analyzed (52 contracts with 84 detectors), 175 result(s) found
```

Slither log >> NonfungibleTokenPositionDescriptor.sol

```
NonfungibleTokenPositionDescriptor.constructor(address,bytes32)._WETH9 (NonfungibleTokenPositionDescriptor.sol#1211) lacks a z
ero-check on :
                - WETH9 = _WETH9 (NonfungibleTokenPositionDescriptor.sol#1212)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

TickMath.getTickAtSqrtRatio(uint160) (NonfungibleTokenPositionDescriptor.sol#596-738) uses assembly
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#603-607)
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#608-612)
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#613-617)
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#618-622)
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#623-627)
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#628-632)
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#633-637)
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#638-641)
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#648-653)
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#654-659)
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#660-665)
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#666-671)
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#672-677)
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#678-683)
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#684-689)
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#690-695)
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#696-701)
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#702-707)
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#708-713)
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#714-719)
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#720-725)
        - INLINE ASM (NonfungibleTokenPositionDescriptor.sol#726-730)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Pragma version=0.8.0 (NonfungibleTokenPositionDescriptor.sol#2) allows old versions
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function IPeripheryImmutableState.WETH9() (NonfungibleTokenPositionDescriptor.sol#75) is not in mixedCase
Function IERC721Permit.PERMIT_TYPEHASH() (NonfungibleTokenPositionDescriptor.sol#136) is not in mixedCase
Function IERC721Permit.DOMAIN_SEPARATOR() (NonfungibleTokenPositionDescriptor.sol#138) is not in mixedCase
Constant NFTDescriptor.sqrt10X128 (NonfungibleTokenPositionDescriptor.sol#859) is not in UPPER_CASE_WITH_UNDERSCORES
Variable NonfungibleTokenPositionDescriptor.WETH9 (NonfungibleTokenPositionDescriptor.sol#1207) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
TickMath.getSqrtRatioAtTick(int24) (NonfungibleTokenPositionDescriptor.sol#566-594) uses literals with too many digits:
        - ratio = 0x100000000000000000000000000000000 (NonfungibleTokenPositionDescriptor.sol#570)
NFTDescriptor.sigfigsRounded(uint256,uint8) (NonfungibleTokenPositionDescriptor.sol#1050-1066) uses literals with too many dig
its:
        - value == 100000 (NonfungibleTokenPositionDescriptor.sol#1061)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

PoolAddress.POOL_INIT_CODE_HASH (NonfungibleTokenPositionDescriptor.sol#45) is never used in PoolAddress (NonfungibleTokenPosi
tionDescriptor.sol#44-69)
TickMath.MAX_TICK (NonfungibleTokenPositionDescriptor.sol#561) is never used in TickMath (NonfungibleTokenPositionDescriptor.s
ol#559-739)
NFTDescriptor.sqrt10X128 (NonfungibleTokenPositionDescriptor.sol#859) is never used in NFTDescriptor (NonfungibleTokenPosition
Descriptor.sol#850-1196)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
NonfungibleTokenPositionDescriptor.sol analyzed (28 contracts with 84 detectors), 67 result(s) found
```

Slither log >> Weav3Factory.sol

```
Weav3Factory.setEnhancedSwapFee(bool,uint24,address).receiver (Weav3Factory.sol#2508) lacks a zero-check on :
                - receiverEnhancedSwapFee = receiver (Weav3Factory.sol#2512)
Weav3Factory.setOwner(address)._owner (Weav3Factory.sol#2516) lacks a zero-check on :
                - owner = _owner (Weav3Factory.sol#2519)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Variable 'TickBitmap.nextInitializedTickWithinOneWord(mapping(int16 => uint256),int24,int24,bool).bitPos (Weav3Factory.sol#827
)' in TickBitmap.nextInitializedTickWithinOneWord(mapping(int16 => uint256),int24,int24,bool) (Weav3Factory.sol#817-845) poten
tially used before declaration: (wordPos,bitPos) = position(compressed + 1) (Weav3Factory.sol#836)
Variable 'TickBitmap.nextInitializedTickWithinOneWord(mapping(int16 => uint256),int24,int24,bool).wordPos (Weav3Factory.sol#82
7)' in TickBitmap.nextInitializedTickWithinOneWord(mapping(int16 => uint256),int24,int24,bool) (Weav3Factory.sol#817-845) pote
ntially used before declaration: (wordPos,bitPos) = position(compressed + 1) (Weav3Factory.sol#836)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in Weav3Pool.flash(address,uint256,uint256,bytes) (Weav3Factory.sol#2346-2389):
        External calls:
        - TransferHelper.safeTransfer(token0,recipient,amount0) (Weav3Factory.sol#2360)
        - TransferHelper.safeTransfer(token1,recipient,amount1) (Weav3Factory.sol#2361)
        - IWeav3FlashCallback(msg.sender).weav3FlashCallback(fee0,fee1,data) (Weav3Factory.sol#2363)
        State variables written after the call(s):
        - feeGrowthGlobal0X128 += FullMath.mulDiv(paid0 - fees0,FixedPoint128.Q128,_liquidity) (Weav3Factory.sol#2379)
        - feeGrowthGlobal1X128 += FullMath.mulDiv(paid1 - fees1,FixedPoint128.Q128,_liquidity) (Weav3Factory.sol#2385)
        - protocolFees.token0 += uint128(fees0) (Weav3Factory.sol#2378)
        - protocolFees.token1 += uint128(fees1) (Weav3Factory.sol#2384)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
FixedPoint96.slitherConstructorConstantVariables() (Weav3Factory.sol#78-81) uses literals with too many digits:
        - Q96 = 0x1000000000000000000000000 (Weav3Factory.sol#80)
FixedPoint128.slitherConstructorConstantVariables() (Weav3Factory.sol#657-659) uses literals with too many digits:
        - Q128 = 0x100000000000000000000000000000000 (Weav3Factory.sol#658)
BitMath.mostSignificantBit(uint256) (Weav3Factory.sol#724-756) uses literals with too many digits:
        - x >= 0x100000000000000000000000000000000 (Weav3Factory.sol#727)
BitMath.mostSignificantBit(uint256) (Weav3Factory.sol#724-756) uses literals with too many digits:
        - x >= 0x10000000000000000 (Weav3Factory.sol#731)
BitMath.mostSignificantBit(uint256) (Weav3Factory.sol#724-756) uses literals with too many digits:
        - x >= 0x100000000 (Weav3Factory.sol#735)
TickMath.getSqrtRatioAtTick(int24) (Weav3Factory.sol#856-884) uses literals with too many digits:
        - ratio = 0x100000000000000000000000000000000 (Weav3Factory.sol#860)
Weav3Factory.enableFeeAmount(uint24,int24) (Weav3Factory.sol#2523-2534) uses literals with too many digits:
        - require(bool)(fee < 1000000) (Weav3Factory.sol#2525)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

TickMath.MAX_TICK (Weav3Factory.sol#851) is never used in TickMath (Weav3Factory.sol#849-1029)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
Weav3Factory.sol analyzed (33 contracts with 84 detectors), 121 result(s) found
```

Slither log >> Quoter.sol

```
Variable 'Quoter.quoteExactInputSingle(address,address,uint24,uint256,uint160).reason (Quoter.sol#771)' in Quoter.quoteExactIn
putSingle(address,address,uint24,uint256,uint160) (Quoter.sol#752-774) potentially used before declaration: parseRevertReason(
reason) (Quoter.sol#772)
Variable 'Quoter.quoteExactOutputSingle(address,address,uint24,uint256,uint160).reason (Quoter.sol#817)' in Quoter.quoteExactO
utputSingle(address,address,uint24,uint256,uint160) (Quoter.sol#796-821) potentially used before declaration: parseRevertReaso
n(reason) (Quoter.sol#819)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in Quoter.quoteExactOutputSingle(address,address,uint24,uint256,uint160) (Quoter.sol#796-821):
        External calls:
        - getPool(tokenIn,tokenOut,fee).swap(address(this),zeroForOne,- amountOut.toInt256(),sqrtPriceLimitX96,abi.encodePacke
d(tokenOut,fee,tokenIn)) (Quoter.sol#807-820)
        - getPool(tokenIn,tokenOut,fee).swap(address(this),zeroForOne,- amountOut.toInt256(),TickMath.MIN_SQRT_RATIO + 1,abi.e
ncodePacked(tokenOut,fee,tokenIn)) (Quoter.sol#807-820)
        - getPool(tokenIn,tokenOut,fee).swap(address(this),zeroForOne,- amountOut.toInt256(),TickMath.MAX_SQRT_RATIO - 1,abi.e
ncodePacked(tokenOut,fee,tokenIn)) (Quoter.sol#807-820)
        State variables written after the call(s):
        - delete amountOutCached (Quoter.sol#818)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
Reentrancy in Quoter.quoteExactOutputSingle(address,address,uint24,uint256,uint160) (Quoter.sol#796-821):
        External calls:
        - getPool(tokenIn,tokenOut,fee).swap(address(this),zeroForOne,- amountOut.toInt256(),sqrtPriceLimitX96,abi.encodePacke
d(tokenOut,fee,tokenIn)) (Quoter.sol#807-820)
        - getPool(tokenIn,tokenOut,fee).swap(address(this),zeroForOne,- amountOut.toInt256(),TickMath.MIN_SQRT_RATIO + 1,abi.e
ncodePacked(tokenOut,fee,tokenIn)) (Quoter.sol#807-820)
        - getPool(tokenIn,tokenOut,fee).swap(address(this),zeroForOne,- amountOut.toInt256(),TickMath.MAX_SQRT_RATIO - 1,abi.e
ncodePacked(tokenOut,fee,tokenIn)) (Quoter.sol#807-820)
        State variables written after the call(s):
        - delete amountOutCached (Quoter.sol#818)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
Quoter.weav3SwapCallback(int256,int256,bytes,bool) (Quoter.sol#708-737) uses assembly
        - INLINE ASM (Quoter.sol#723-727)
        - INLINE ASM (Quoter.sol#731-735)
Quoter.parseRevertReason(bytes) (Quoter.sol#740-749) uses assembly
        - INLINE ASM (Quoter.sol#743-745)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Quoter.quoteExactOutputSingle(address,address,uint24,uint256,uint160) (Quoter.sol#796-821) has costly operations inside a loop
:
        - amountOutCached = amountOut (Quoter.sol#806)
Quoter.quoteExactOutputSingle(address,address,uint24,uint256,uint160) (Quoter.sol#796-821) has costly operations inside a loop
:
        - delete amountOutCached (Quoter.sol#818)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
```

```
BytesLib.toAddress(bytes,uint256) (Quoter.sol#415-425) uses literals with too many digits:
        - tempAddress = mload(uint256)(_bytes + 0x20 + _start) / 0x1000000000000000000000000 (Quoter.sol#421)
TickMath.getSqrtRatioAtTick(int24) (Quoter.sol#495-523) uses literals with too many digits:
        - ratio = 0x100000000000000000000000000000000 (Quoter.sol#499)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
Quoter.sol analyzed (19 contracts with 84 detectors), 54 result(s) found
```

Slither log >> TickLens.sol

```
TickLens.getPopulatedTicksInWord(address,int16) (TickLens.sol#252-281) has external calls inside a loop: (liquidityGross,liqui
dityNet) = IWeav3Pool(pool).ticks(populatedTick) (TickLens.sol#273)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Pragma version>=0.5.0 (TickLens.sol#2) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Variable IWeav3PoolState.positions(bytes32).feeGrowthInside0LastX128 (TickLens.sol#63) is too similar to IWeav3PoolState.posit
ions(bytes32).feeGrowthInside1LastX128 (TickLens.sol#64)
Variable IWeav3PoolState.ticks(int24).feeGrowthOutside0X128 (TickLens.sol#48) is too similar to IWeav3PoolState.ticks(int24).f
eeGrowthOutside1X128 (TickLens.sol#49)
Variable IWeav3PoolState.positions(bytes32).tokensOwed0 (TickLens.sol#65) is too similar to IWeav3PoolState.positions(bytes32)
.tokensOwed1 (TickLens.sol#66)
Variable IWeav3PoolActions.collect(address,int24,int24,uint128,uint128).amount0Requested (TickLens.sol#97) is too similar to I
Weav3PoolActions.collect(address,int24,int24,uint128,uint128).amount1Requested (TickLens.sol#98)
Variable IWeav3PoolOwnerActions.collectProtocol(address,uint128,uint128).amount0Requested (TickLens.sol#140) is too similar to
 IWeav3PoolOwnerActions.collectProtocol(address,uint128,uint128).amount1Requested (TickLens.sol#141)
Variable IWeav3PoolOwnerActions.setFeeProtocol(uint8,uint8).feeProtocol0 (TickLens.sol#136) is too similar to IWeav3PoolOwnerA
ctions.setFeeProtocol(uint8,uint8).feeProtocol1 (TickLens.sol#136)
```

```
ctions.setFeeProtocol(uint8,uint8).feeProtocol1 (TickLens.sol#136)
Variable IWeav3PoolActions.collect(address,int24,int24,uint128,uint128).amount0Requested (TickLens.sol#97) is too similar to I
Weav3PoolOwnerActions.collectProtocol(address,uint128,uint128).amount1Requested (TickLens.sol#141)
Variable IWeav3PoolOwnerActions.collectProtocol(address,uint128,uint128).amount0Requested (TickLens.sol#140) is too similar to
 IWeav3PoolActions.collect(address,int24,int24,uint128,uint128).amount1Requested (TickLens.sol#98)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
TickLens.sol analyzed (9 contracts with 84 detectors), 11 result(s) found
```

Slither log >> Multicall.sol

```
Multicall.multicall(bytes[]) (Multicall.sol#11-26) has external calls inside a loop: (success,result) = address(this).delegate
call(data[i]) (Multicall.sol#14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Multicall.multicall(bytes[]) (Multicall.sol#11-26) uses assembly
        - INLINE ASM (Multicall.sol#18-20)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Low level call in Multicall.multicall(bytes[]) (Multicall.sol#11-26):
        - (success,result) = address(this).delegatecall(data[i]) (Multicall.sol#14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
Multicall.sol analyzed (2 contracts with 84 detectors), 4 result(s) found
```

Slither log >> Weav3Token.sol

```
Weav3Token.setMinter(address)._minter (Weav3Token.sol#46) lacks a zero-check on :
                - minter = _minter (Weav3Token.sol#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Pragma version^0.8.0 (Weav3Token.sol#2) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter Weav3Token.setMinter(address)._minter (Weav3Token.sol#46) is not in mixedCase
Parameter Weav3Token.initialMint(address)._recipient (Weav3Token.sol#52) is not in mixedCase
Parameter Weav3Token.approve(address,uint256)._spender (Weav3Token.sol#58) is not in mixedCase
Parameter Weav3Token.approve(address,uint256)._value (Weav3Token.sol#58) is not in mixedCase
Parameter Weav3Token.transfer(address,uint256)._to (Weav3Token.sol#84) is not in mixedCase
Parameter Weav3Token.transfer(address,uint256)._value (Weav3Token.sol#84) is not in mixedCase
Parameter Weav3Token.transferFrom(address,address,uint256)._from (Weav3Token.sol#89) is not in mixedCase
Parameter Weav3Token.transferFrom(address,address,uint256)._to (Weav3Token.sol#90) is not in mixedCase
Parameter Weav3Token.transferFrom(address,address,uint256)._value (Weav3Token.sol#91) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Weav3Token.constructor() (Weav3Token.sol#40-43) uses literals with too many digits:
        - _mint(msg.sender,10000000000000000000000) (Weav3Token.sol#42)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
Weav3Token.sol analyzed (2 contracts with 84 detectors), 13 result(s) found
```

Slither log >> DeployUniversalRouter.sol

```
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (DeployUniversalRoute
r.sol#976)' in ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (DeployUniversalRouter.so
l#967-986) potentially used before declaration: response != IERC1155Receiver.onERC1155Received.selector (DeployUniversalRouter
.sol#977)
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (DeployUniversalRouter.
sol#980)' in ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (DeployUniversalRouter.sol#
967-986) potentially used before declaration: revert(string)(reason) (DeployUniversalRouter.sol#981)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (DeployUnive
rsalRouter.sol#998)' in ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (Deploy
UniversalRouter.sol#988-1009) potentially used before declaration: response != IERC1155Receiver.onERC1155BatchReceived.selecto
r (DeployUniversalRouter.sol#1000)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (DeployUnivers
alRouter.sol#1003)' in ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (DeployU
niversalRouter.sol#988-1009) potentially used before declaration: revert(string)(reason) (DeployUniversalRouter.sol#1004)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (DeployUniversalRouter.sol#1270)' in ERC721._che
ckOnERC721Received(address,address,uint256,bytes) (DeployUniversalRouter.sol#1263-1284) potentially used before declaration: r
etval == IERC721Receiver.onERC721Received.selector (DeployUniversalRouter.sol#1271)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (DeployUniversalRouter.sol#1272)' in ERC721._che
ckOnERC721Received(address,address,uint256,bytes) (DeployUniversalRouter.sol#1263-1284) potentially used before declaration: r
eason.length == 0 (DeployUniversalRouter.sol#1273)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (DeployUniversalRouter.sol#1272)' in ERC721._che
ckOnERC721Received(address,address,uint256,bytes) (DeployUniversalRouter.sol#1263-1284) potentially used before declaration: r
evert(uint256,uint256)(32 + reason,mload(uint256)(reason)) (DeployUniversalRouter.sol#1277)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in RewardsCollector.collectRewards(bytes) (DeployUniversalRouter.sol#460-465):
        External calls:
        - LOOKS_RARE_TOKEN.transfer(ROUTER_REWARDS_DISTRIBUTOR,balance) (DeployUniversalRouter.sol#463)
        Event emitted after the call(s):
        - RewardsSent(balance) (DeployUniversalRouter.sol#464)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
Pragma version^0.8.17 (DeployUniversalRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0
.6.12/0.7.6/0.8.16
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (DeployUniversalRouter.sol#131-136):
        - (success) = recipient.call{value: amount}() (DeployUniversalRouter.sol#134)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (DeployUniversalRouter.sol#168-190):
        - (success,returndata) = target.call{value: weiValue}(data) (DeployUniversalRouter.sol#176)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable RouterImmutables.WETH9 (DeployUniversalRouter.sol#387) is not in mixedCase
Variable RouterImmutables.SEAPORT_V1_5 (DeployUniversalRouter.sol#391) is not in mixedCase
Variable RouterImmutables.SEAPORT_V1_4 (DeployUniversalRouter.sol#393) is not in mixedCase
Variable RouterImmutables.OPENSEA_CONDUIT (DeployUniversalRouter.sol#395) is not in mixedCase
Variable RouterImmutables.NFTX_ZAP (DeployUniversalRouter.sol#397) is not in mixedCase
Variable RouterImmutables.X2Y2 (DeployUniversalRouter.sol#399) is not in mixedCase
Variable RouterImmutables.FOUNDATION (DeployUniversalRouter.sol#401) is not in mixedCase
Variable RouterImmutables.SUDOSWAP (DeployUniversalRouter.sol#403) is not in mixedCase
Variable RouterImmutables.ELEMENT_MARKET (DeployUniversalRouter.sol#405) is not in mixedCase
Variable RouterImmutables.NFT20_ZAP (DeployUniversalRouter.sol#407) is not in mixedCase
Variable RouterImmutables.CRYPTOPUNKS (DeployUniversalRouter.sol#409) is not in mixedCase
Variable RouterImmutables.LOOKS_RARE_V2 (DeployUniversalRouter.sol#411) is not in mixedCase
Variable RouterImmutables.LOOKS_RARE_TOKEN (DeployUniversalRouter.sol#413) is not in mixedCase
Variable RouterImmutables.LOOKS_RARE_REWARDS_DISTRIBUTOR (DeployUniversalRouter.sol#415) is not in mixedCase
Variable RouterImmutables.ROUTER_REWARDS_DISTRIBUTOR (DeployUniversalRouter.sol#417) is not in mixedCase
Variable RouterImmutables.UNISWAP_V2_FACTORY (DeployUniversalRouter.sol#419) is not in mixedCase
Variable RouterImmutables.UNISWAP_V2_PAIR_INIT_CODE_HASH (DeployUniversalRouter.sol#421) is not in mixedCase
Variable RouterImmutables.UNISWAP_V3_FACTORY (DeployUniversalRouter.sol#423) is not in mixedCase
Variable RouterImmutables.UNISWAP_V3_POOL_INIT_CODE_HASH (DeployUniversalRouter.sol#425) is not in mixedCase
Function ERC721.__unsafe_increaseBalance(address,uint256) (DeployUniversalRouter.sol#1290-1292) is not in mixedCase
Parameter BytesLib.slice(bytes,uint256,uint256)._bytes (DeployUniversalRouter.sol#1650) is not in mixedCase
Parameter BytesLib.slice(bytes,uint256,uint256)._start (DeployUniversalRouter.sol#1651) is not in mixedCase
```

```
Variable Weav3SwapRouter._swap(int256,address,bytes,address,bool,bool).amount0Delta (DeployUniversalRouter.sol#1824) is too si
milar to Weav3SwapRouter.SwapExactOutput(address,uint256,uint256,bytes,address,bool).amount1Delta (DeployUniversalRouter.sol#1
811)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

Constants.slitherConstructorConstantVariables() (DeployUniversalRouter.sol#618-638) uses literals with too many digits:
        - CONTRACT_BALANCE = 0x8000000000000000000000000000000000000000000000000000000000000000 (DeployUniversalRouter.sol#619
)
BytesLib.toAddress(bytes,uint256) (DeployUniversalRouter.sol#1694-1704) uses literals with too many digits:
        - tempAddress = mload(uint256)(_bytes + 0x20 + _start) / 0x1000000000000000000000000 (DeployUniversalRouter.sol#1700)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Payments.FEE_BIPS_BASE (DeployUniversalRouter.sol#1297) is never used in V2SwapRouter (DeployUniversalRouter.sol#1987-2052)
Payments.FEE_BIPS_BASE (DeployUniversalRouter.sol#1297) is never used in UniversalRouter (DeployUniversalRouter.sol#2099-2141)
Weav3SwapRouter.MIN_SQRT_RATIO (DeployUniversalRouter.sol#1772) is never used in UniversalRouter (DeployUniversalRouter.sol#20
99-2141)
Weav3SwapRouter.MAX_SQRT_RATIO (DeployUniversalRouter.sol#1774) is never used in UniversalRouter (DeployUniversalRouter.sol#20
99-2141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

ERC20._name (DeployUniversalRouter.sol#218) should be immutable
ERC20._symbol (DeployUniversalRouter.sol#219) should be immutable
ERC721._name (DeployUniversalRouter.sol#1064) should be immutable
ERC721._symbol (DeployUniversalRouter.sol#1066) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
DeployUniversalRouter.sol analyzed (52 contracts with 84 detectors), 165 result(s) found
```

Slither log >> UniversalRouter.sol

```
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (UniversalRout
er.sol#1242)' in ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (UniversalRout
er.sol#1227-1248) potentially used before declaration: revert(string)(reason) (UniversalRouter.sol#1243)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (UniversalRouter.sol#1685)' in ERC721._checkOnER
C721Received(address,address,uint256,bytes) (UniversalRouter.sol#1678-1700) potentially used before declaration: retval == IER
C721Receiver.onERC721Received.selector (UniversalRouter.sol#1686)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (UniversalRouter.sol#1687)' in ERC721._checkOnER
C721Received(address,address,uint256,bytes) (UniversalRouter.sol#1678-1700) potentially used before declaration: reason.length
 == 0 (UniversalRouter.sol#1688)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (UniversalRouter.sol#1687)' in ERC721._checkOnER
C721Received(address,address,uint256,bytes) (UniversalRouter.sol#1678-1700) potentially used before declaration: revert(uint25
6,uint256)(32 + reason,mload(uint256)(reason)) (UniversalRouter.sol#1693)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in RewardsCollector.collectRewards(bytes) (UniversalRouter.sol#691-698):
        External calls:
        - (success) = LOOKS_RARE_REWARDS_DISTRIBUTOR.call(looksRareClaim) (UniversalRouter.sol#692)
        - LOOKS_RARE_TOKEN.transfer(ROUTER_REWARDS_DISTRIBUTOR,balance) (UniversalRouter.sol#696)
        Event emitted after the call(s):
        - RewardsSent(balance) (UniversalRouter.sol#697)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
Pragma version^0.8.17 (UniversalRouter.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/
0.7.6/0.8.16
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (UniversalRouter.sol#141-146):
        - (success) = recipient.call{value: amount}() (UniversalRouter.sol#144)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (UniversalRouter.sol#178-200):
        - (success,returndata) = target.call{value: weiValue}(data) (UniversalRouter.sol#186)
Low level call in RewardsCollector.collectRewards(bytes) (UniversalRouter.sol#691-698):
        - (success) = LOOKS_RARE_REWARDS_DISTRIBUTOR.call(looksRareClaim) (UniversalRouter.sol#692)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable RouterImmutables.WETH9 (UniversalRouter.sol#593) is not in mixedCase
Variable RouterImmutables.SEAPORT_V1_5 (UniversalRouter.sol#599) is not in mixedCase
Variable RouterImmutables.SEAPORT_V1_4 (UniversalRouter.sol#602) is not in mixedCase
Variable RouterImmutables.OPENSEA_CONDUIT (UniversalRouter.sol#605) is not in mixedCase
Variable RouterImmutables.NFTX_ZAP (UniversalRouter.sol#608) is not in mixedCase
Variable RouterImmutables.X2Y2 (UniversalRouter.sol#611) is not in mixedCase
Variable RouterImmutables.FOUNDATION (UniversalRouter.sol#614) is not in mixedCase
Variable RouterImmutables.SUDOSWAP (UniversalRouter.sol#617) is not in mixedCase
Variable RouterImmutables.ELEMENT_MARKET (UniversalRouter.sol#620) is not in mixedCase
Variable RouterImmutables.NFT20_ZAP (UniversalRouter.sol#623) is not in mixedCase
Variable RouterImmutables.CRYPTOPUNKS (UniversalRouter.sol#626) is not in mixedCase
Variable RouterImmutables.LOOKS_RARE_V2 (UniversalRouter.sol#629) is not in mixedCase
Variable RouterImmutables.LOOKS_RARE_TOKEN (UniversalRouter.sol#632) is not in mixedCase
Variable RouterImmutables.LOOKS_RARE_REWARDS_DISTRIBUTOR (UniversalRouter.sol#635) is not in mixedCase
Variable RouterImmutables.ROUTER_REWARDS_DISTRIBUTOR (UniversalRouter.sol#638) is not in mixedCase
Variable RouterImmutables.UNISWAP_V2_FACTORY (UniversalRouter.sol#641) is not in mixedCase
Variable RouterImmutables.UNISWAP_V2_PAIR_INIT_CODE_HASH (UniversalRouter.sol#644) is not in mixedCase
Variable RouterImmutables.UNISWAP_V3_FACTORY (UniversalRouter.sol#647) is not in mixedCase
```

```
Function Weav3SwapRouter.SwapExactInput(address,uint256,uint256,bytes,address,bool) (UniversalRouter.sol#2317-2326) is not in
mixedCase
Function Weav3SwapRouter.SwapExactOutput(address,uint256,uint256,bytes,address,bool) (UniversalRouter.sol#2334-2353) is not in
 mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (UniversalRouter.sol#2403) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (UniversalRouter.sol#2404) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (UniversalRouter.sol#2421) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
Constants.slitherConstructorConstantVariables() (UniversalRouter.sol#856-876) uses literals with too many digits:
        - CONTRACT_BALANCE = 0x8000000000000000000000000000000000000000000000000000000000000000 (UniversalRouter.sol#857)
BytesLib.toAddress(bytes,uint256) (UniversalRouter.sol#2217-2227) uses literals with too many digits:
        - tempAddress = mload(uint256)(_bytes + 0x20 + _start) / 0x1000000000000000000000000 (UniversalRouter.sol#2223)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
```

```
Payments.FEE_BIPS_BASE (UniversalRouter.sol#1750) is never used in V2SwapRouter (UniversalRouter.sol#2540-2621)
Payments.FEE_BIPS_BASE (UniversalRouter.sol#1750) is never used in UniversalRouter (UniversalRouter.sol#2684-2732)
Weav3SwapRouter.MIN_SQRT_RATIO (UniversalRouter.sol#2300) is never used in UniversalRouter (UniversalRouter.sol#2684-2732)
Weav3SwapRouter.MAX_SQRT_RATIO (UniversalRouter.sol#2303) is never used in UniversalRouter (UniversalRouter.sol#2684-2732)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
```

```
ERC20._name (UniversalRouter.sol#238) should be immutable
ERC20._symbol (UniversalRouter.sol#239) should be immutable
ERC721._name (UniversalRouter.sol#1303) should be immutable
ERC721._symbol (UniversalRouter.sol#1306) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
UniversalRouter.sol analyzed (49 contracts with 84 detectors), 163 result(s) found
```

Slither log >> Permit2.sol

```
ERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (Permit2.sol#622-666) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,PERMIT_DEADLINE_EXPIRED) (Permit2.sol#631)
AllowanceTransfer.permit(address,IAllowanceTransfer.PermitSingle,bytes) (Permit2.sol#889-896) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > permitSingle.sigDeadline (Permit2.sol#890)
AllowanceTransfer.permit(address,IAllowanceTransfer.PermitBatch,bytes) (Permit2.sol#899-912) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > permitBatch.sigDeadline (Permit2.sol#900)
AllowanceTransfer._transfer(address,address,uint160,address) (Permit2.sol#932-950) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > allowed.expiration (Permit2.sol#935)
SignatureTransfer._permitTransferFrom(ISignatureTransfer.PermitTransferFrom,ISignatureTransfer.SignatureTransferDetails,addres
s,bytes32,bytes) (Permit2.sol#1039-1056) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > permit.deadline (Permit2.sol#1048)
SignatureTransfer._permitTransferFrom(ISignatureTransfer.PermitBatchTransferFrom,ISignatureTransfer.SignatureTransferDetails[]
,address,bytes32,bytes) (Permit2.sol#1085-1113) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > permit.deadline (Permit2.sol#1094)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Allowance.updateAll(IAllowanceTransfer.PackedAllowance,uint160,uint48,uint48) (Permit2.sol#319-336) uses assembly
        - INLINE ASM (Permit2.sol#333-335)
SafeTransferLib.safeTransferETH(address,uint256) (Permit2.sol#720-730) uses assembly
        - INLINE ASM (Permit2.sol#724-727)
SafeTransferLib.safeTransferFrom(ERC20,address,address,uint256) (Permit2.sol#736-768) uses assembly
        - INLINE ASM (Permit2.sol#745-765)
SafeTransferLib.safeTransfer(ERC20,address,uint256) (Permit2.sol#770-800) uses assembly
        - INLINE ASM (Permit2.sol#778-797)
SafeTransferLib.safeApprove(ERC20,address,uint256) (Permit2.sol#802-832) uses assembly
        - INLINE ASM (Permit2.sol#810-829)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
ERC20._burn(address,uint256) (Permit2.sol#701-711) is never used and should be removed
ERC20._mint(address,uint256) (Permit2.sol#689-699) is never used and should be removed
PermitHash.hashWithWitness(ISignatureTransfer.PermitBatchTransferFrom,bytes32,string) (Permit2.sol#446-471) is never used and
should be removed
SafeTransferLib.safeApprove(ERC20,address,uint256) (Permit2.sol#802-832) is never used and should be removed
SafeTransferLib.safeTransfer(ERC20,address,uint256) (Permit2.sol#770-800) is never used and should be removed
SafeTransferLib.safeTransferETH(address,uint256) (Permit2.sol#720-730) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.17 (Permit2.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8
.16
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

ERC20 (Permit2.sol#529-712) should inherit from IEIP712 (Permit2.sol#22-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance

Function IEIP712.DOMAIN_SEPARATOR() (Permit2.sol#23) is not in mixedCase
Function ERC20.DOMAIN_SEPARATOR() (Permit2.sol#668-670) is not in mixedCase
Variable ERC20.INITIAL_CHAIN_ID (Permit2.sol#552) is not in mixedCase
Variable ERC20.INITIAL_DOMAIN_SEPARATOR (Permit2.sol#554) is not in mixedCase
Function EIP712.DOMAIN_SEPARATOR() (Permit2.sol#852-856) is not in mixedCase
Variable EIP712._CACHED_DOMAIN_SEPARATOR (Permit2.sol#838) is not in mixedCase
Variable EIP712._CACHED_CHAIN_ID (Permit2.sol#839) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

SafeTransferLib.safeTransferFrom(ERC20,address,address,uint256) (Permit2.sol#736-768) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransferFrom_asm_0,0x23b872dd0000000000000000000000000000000000000000
000000000000000) (Permit2.sol#750)
SafeTransferLib.safeTransfer(ERC20,address,uint256) (Permit2.sol#770-800) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeTransfer_asm_0,0xa9059cbb0000000000000000000000000000000000000000
00000000000) (Permit2.sol#783)
SafeTransferLib.safeApprove(ERC20,address,uint256) (Permit2.sol#802-832) uses literals with too many digits:
        - mstore(uint256,uint256)(freeMemoryPointer_safeApprove_asm_0,0x095ea7b3000000000000000000000000000000000000000000
0000000000) (Permit2.sol#815)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
```

```
ERC20.name (Permit2.sol#537) should be immutable
ERC20.symbol (Permit2.sol#539) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
Permit2.sol analyzed (13 contracts with 84 detectors), 38 result(s) found
```

Solidity Static Analysis

NonfungiblePositionManager.sol

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 463:20:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 10:15:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.
more
Pos: 57:27:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 58:8:

Similar variable names:

NonfungiblePositionManager.mint(struct INonfungiblePositionManager.MintParams) : Variables have very similar names "pool" and "poolId". Note: Modifiers are currently not considered by this static analysis.
Pos: 745:20:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 957:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 90:12:

NonfungibleTokenPositionDescriptor.sol

Gas costs:

Gas requirement of function NonfungibleTokenPositionDescriptor.flipRatio is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 35:4:

No return:

NonfungibleTokenPositionDescriptor.tokenURI(contract INonfungiblePositionManager,uint256): Defines a return type but never explicitly returns a value.
Pos: 25:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 38:8:

Similar variable names:

PoolAddress.getPoolKey(address,address,uint24) : Variables have very similar names "tokenA" and "tokenB".
Pos: 26:48:

Weav3Factory.sol

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 143:22:

Similar variable names:

Weav3Factory.createPool(address,address,uint24) : Variables have very similar names "token0" and "tokenB". Note: Modifiers are currently not considered by this static analysis.
Pos: 46:9:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 958:52:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 968:8:

Weav3Token.sol

Gas costs:

Gas requirement of function Weav3Token.symbol is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 23:4:

Similar variable names:

Weav3Token.mint(address,uint256) : Variables have very similar names "account" and "amount".
Pos: 99:23:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 98:8:

Quoter.sol

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 75:12:

Similar variable names:

Quoter.weav3SwapCallback(int256,int256,bytes,bool) : Variables have very similar names "amount0Delta" and "amount1Delta".
Pos: 51:12:

Gas costs:

Gas requirement of function Quoter.quoteExactOutput is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 156:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 79:10:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 34:16:

DeployUniversalRouter.sol

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 2092:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 1955:20:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 1955:20:

TickLens.sol

Gas costs:

Gas requirement of function TickLens.getPopulatedTicksInWord is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 12:4:

Similar variable names:

TickLens.getPopulatedTicksInWord(address,int16) : Variables have very similar names "populatedTick" and "populatedTicks".
Pos: 35:26:

No return:

ITickLens.getPopulatedTicksInWord(address,int16): Defines a return type but never explicitly returns a value.
Pos: 21:4:

UniversalRouter.sol

Gas costs:

Gas requirement of function UniversalRouter.execute is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 2693:4:

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type

more

Pos: 2394:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.

more

Pos: 1162:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 2231:8:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.

more

Pos: 1622:8:

Multicall.sol

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 19:16:

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 14:50:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 13:8:

Similar variable names:

Multicall.multicall(bytes[]) : Variables have very similar names "result" and "results".
Pos: 25:25:

No return:

IMulticall.multicall(bytes[]): Defines a return type but never explicitly returns a value.
Pos: 12:4:

Permit2.sol

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 810:8:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1094:12:

Gas costs:

Gas requirement of function AllowanceTransfer.permit is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 899:4:

Gas costs:

Gas requirement of function AllowanceTransfer.transferFrom is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 920:4:

Gas costs:

Gas requirement of function AllowanceTransfer.lockdown is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 953:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.

more

Pos: 1095:8:

Similar variable names:

AllowanceTransfer.transferFrom(struct IAllowanceTransfer.AllowanceTransferDetails[]) : Variables have very similar names "transferDetail" and "transferDetails".

Pos: 925:66:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 831:8:

Solhint Linter

NonfungiblePositionManager.sol

```
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:644
Variable name must be in mixedCase
Pos: 9:646
Use double quotes for string literals
Pos: 58:758
Provide an error message for require
Pos: 9:763
Code contains empty blocks
Pos: 61:768
Error message for require is too long
Pos: 9:966
Use double quotes for string literals
Pos: 35:966
```

NonfungibleTokenPositionDescriptor.sol

```
Compiler version =0.7.6 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
global import of path
@weav3npm/core/contracts/interfaces/IWeav3Pool.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:4
Use double quotes for string literals
Pos: 8:4
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:14
Variable name must be in mixedCase
Pos: 17:14
Code contains empty blocks
Pos: 68:14
Code contains empty blocks
Pos: 72:19
Code contains empty blocks
Pos: 5:29
Variable "token" is unused
Pos: 33:42
Variable "chainId" is unused
Pos: 48:42
```

Weav3Factory.sol

```
Compiler version =0.7.6 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
global import of path
@weav3npm/core/contracts/interfaces/IWeav3Factory.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:25
Provide an error message for require
Pos: 9:78
Provide an error message for require
Pos: 9:79
```

Weav3Token.sol

```
Compiler version ^0.7.6 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:0
Constant name must be in capitalized SNAKE_CASE
Pos: 5:23
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:39
Provide an error message for require
Pos: 9:46
Provide an error message for require
Pos: 9:52
Variable name must be in mixedCase
Pos: 9:89
Provide an error message for require
Pos: 9:97
```

Quoter.sol

```
Compiler version =0.7.6 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
global import of path @weav3npm/core/contracts/libraries/SafeCast.sol
is not allowed. Specify names to import individually or bind all
exports of the module into a name (import "path" as Name)
Pos: 1:4
global import of path PoolAddress.sol is not allowed. Specify names
to import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:13
```

```
Use double quotes for string literals
Pos: 8:13
global import of path CallbackValidation.sol is not allowed. Specify
names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:14
Use double quotes for string literals
Pos: 8:14
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:27
Code contains empty blocks
Pos: 9:102
Code contains empty blocks
Pos: 9:148
```

TickLens.sol

```
Compiler version >=0.5.0 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
global import of path
@weav3npm/core/contracts/interfaces/IWeav3Pool.sol is not allowed.
Specify names to import individually or bind all exports of the
module into a name (import "path" as Name)
Pos: 1:4
Use double quotes for string literals
Pos: 8:4
global import of path ITickLens.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:6
Use double quotes for string literals
Pos: 8:6
```

DeployUniversalRouter.sol

```
Compiler version ^0.8.17 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
Code contains empty blocks
Pos: 1:30
When fallback is not payable you will not be able to receive ether
Pos: 5:2145
Explicitly mark visibility of state
Pos: 5:2154
Explicitly mark visibility of state
Pos: 5:2155
```

UniversalRouter.sol

```
Compiler version ^0.8.17 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
Code contains empty blocks
Pos: 1:30
Code contains empty blocks
Pos: 1:39
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:2689
Code contains empty blocks
Pos: 74:2689
Code contains empty blocks
Pos: 55:2715
Code contains empty blocks
Pos: 32:2730
```

Permit2.sol

```
Compiler version 0.8.17 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
Function name must be in mixedCase
Pos: 5:22
Avoid making time-based decisions in your business logic
Pos: 85:329
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:332
Avoid making time-based decisions in your business logic
Pos: 55:345
Explicitly mark visibility of state
Pos: 5:498
Variable name must be in mixedCase
Pos: 5:551
Variable name must be in mixedCase
Pos: 5:553
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:558
Avoid making time-based decisions in your business logic
Pos: 29:630
Function name must be in mixedCase
Pos: 5:667
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:723
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:744
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:777
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:809
Variable name must be in mixedCase
Pos: 5:837
Variable name must be in mixedCase
```

```
Pos: 5:838
Explicitly mark visibility in function (Set ignoreConstructors to
true if using solidity >=0.7.0)
Pos: 5:844
Function name must be in mixedCase
Pos: 5:851
Avoid making time-based decisions in your business logic
Pos: 13:889
Avoid making time-based decisions in your business logic
Pos: 13:899
Avoid making time-based decisions in your business logic
Pos: 13:934
Avoid making time-based decisions in your business logic
Pos: 13:1047
Code contains empty blocks
Pos: 16:1075
Avoid making time-based decisions in your business logic
Pos: 13:1093
Code contains empty blocks
Pos: 1:1145
```

Multicall.sol

```
Compiler version =0.7.6 does not satisfy the ^0.5.8 semver
requirement
Pos: 1:1
global import of path IMulticall.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name
(import "path" as Name)
Pos: 1:4
Use double quotes for string literals
Pos: 8:4
Avoid using low level calls.
Pos: 51:13
Provide an error message for revert
Pos: 41:17
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 17:18
```

Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

Email: audit@EtherAuthority.io