QuillAudits

Audit Report
February, 2023

For

CrowdSwap

Table of Content

Executive Summary

Project Name CrowdSwap

Overview CrowdSwap is a DeFi ecosystem centred around price routing and aggregation for swapping tokens with cross chain mechanisms. The current audited Opportunities Smart Contracts assist in earning, investing in aspects like yield farming, liquidity mining and staking.

Scope of Audit The scope of this audit was to analyse CrowdSwap Opportunities Contracts codebase for quality, security, and correctness. This included testing of smart contracts to ensure proper logic was followed, manual analysis ,checking for bugs and vulnerabilities, checks for dead code, checks for code style, security and more. The audited contracts are as follows:

Git Repo link: *https://github.com/CrowdSwap/Opportunities*
Git Branch: master branch
Commit Hash: e2793ca83555520b2945cfd57055d1c14bca9df0
Fixed in: *https://github.com/CrowdSwap/Opportunities/commit/393906dbc080e835204ce1fee56e95b5bc4717f2*
Git Branch: master branch
Commit Hash: 393906dbc080e835204ce1fee56e95b5bc4717f2

High Medium Low Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	0	2

Contracts Information

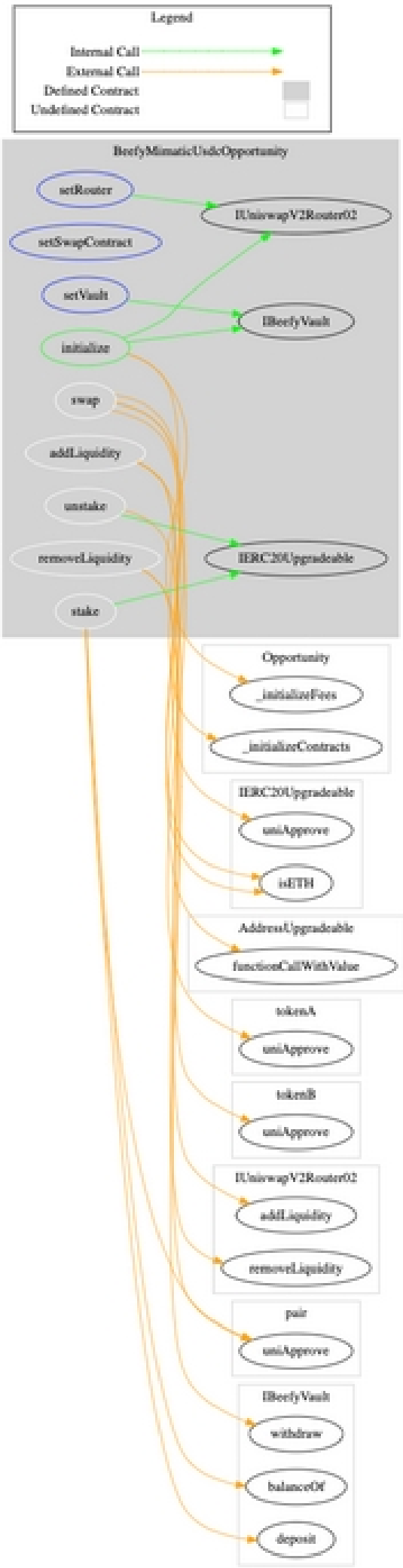
Contract	Lines	Complexity Score
/interfaces/IBeefyVault.sol	9	7
/interfaces/IPancakeMasterChefV2.sol	12	11
/interfaces/IStakingLP.sol	10	7
/interfaces/IUniswapV2Router02.sol	48	14
/helpers/OwnableUpgradeable.sol	83	20
/libraries/UniERC20Upgradeable.sol	48	28
/opportunity/BeefyMimaticUsdcOpportunity.sol	154	61
/opportunity/PancakeOpportunity.sol	297	124
/opportunity/CrowdUsdtLpStakeOpportunity.sol	152	60
/opportunity/Opportunity.sol	472	246
/opportunity/StakingLP.sol	384	185
TOTALS	1669	763

Dependencies

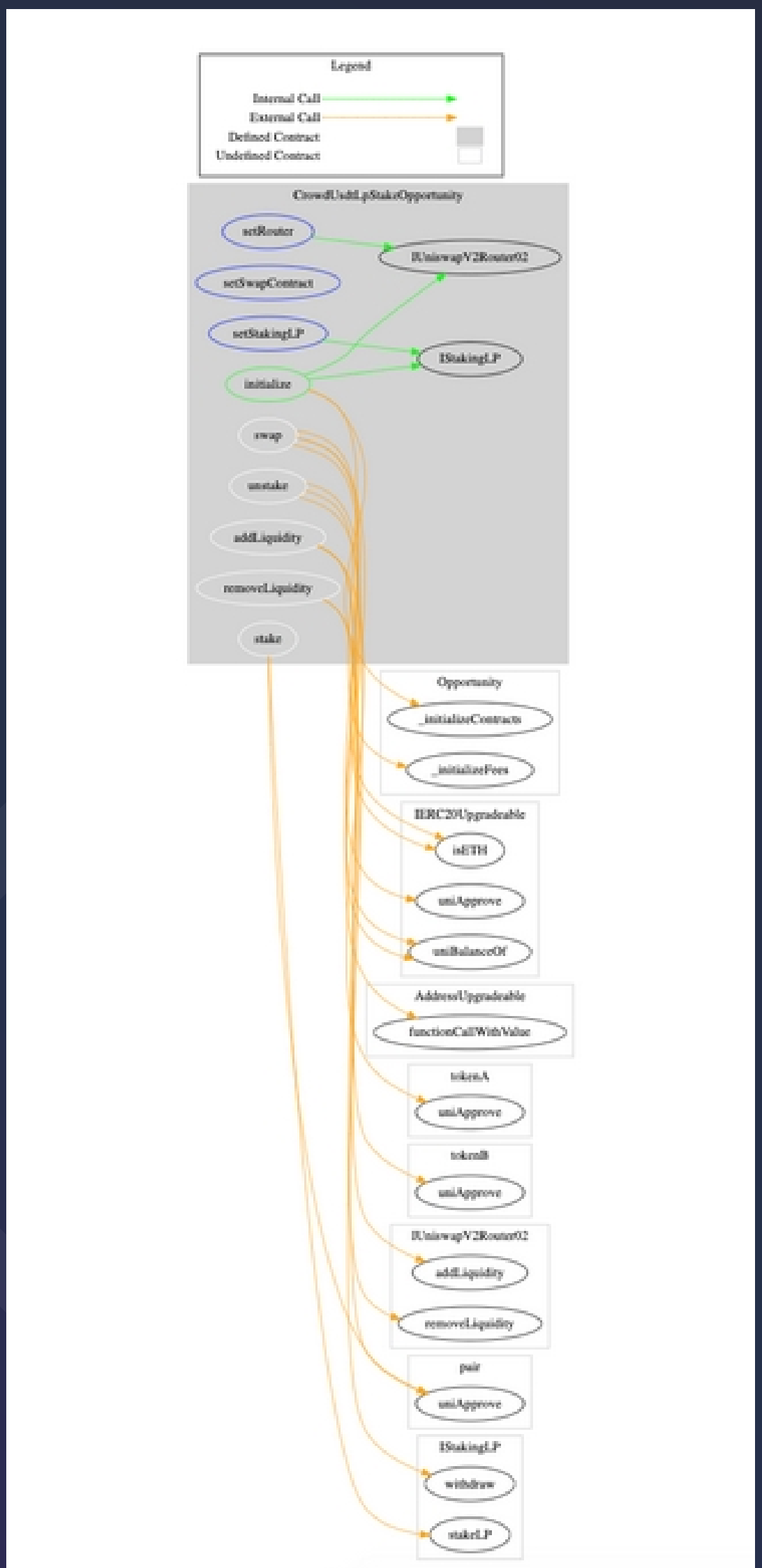
Dependency / Import Path	Count
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	3
@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol	5
@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol	4
@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol	3

Call Graph

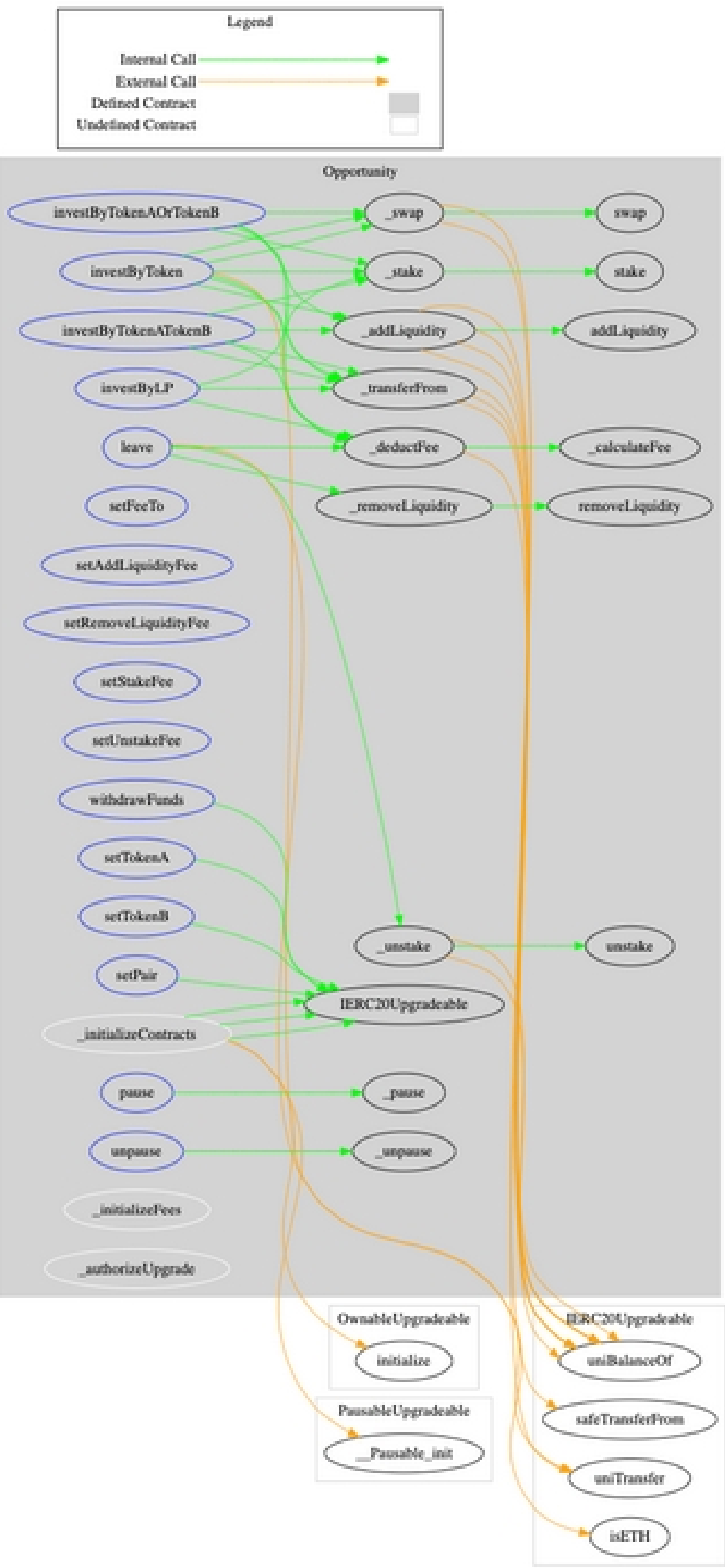
contracts/opportunity/BeefyMimaticUsdcOpportunity.sol



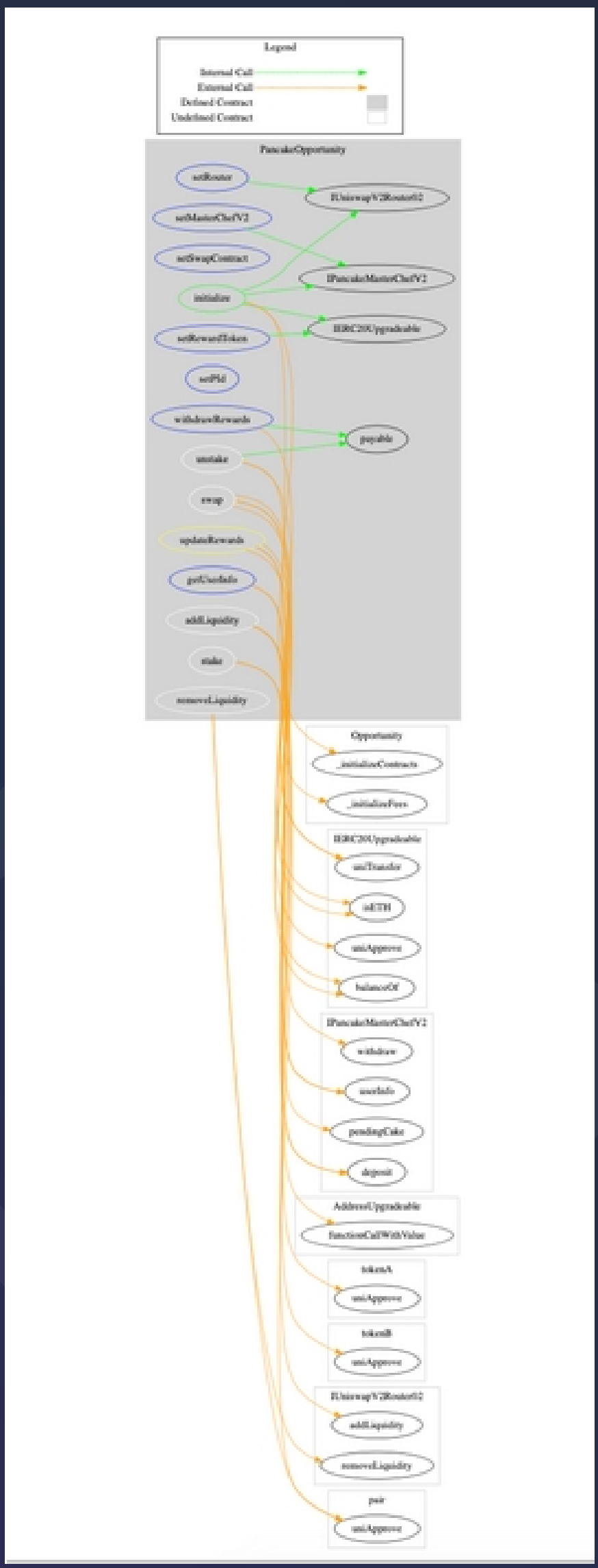
contracts/opportunity/CrowdUsdtLpStakeOpportunity.sol



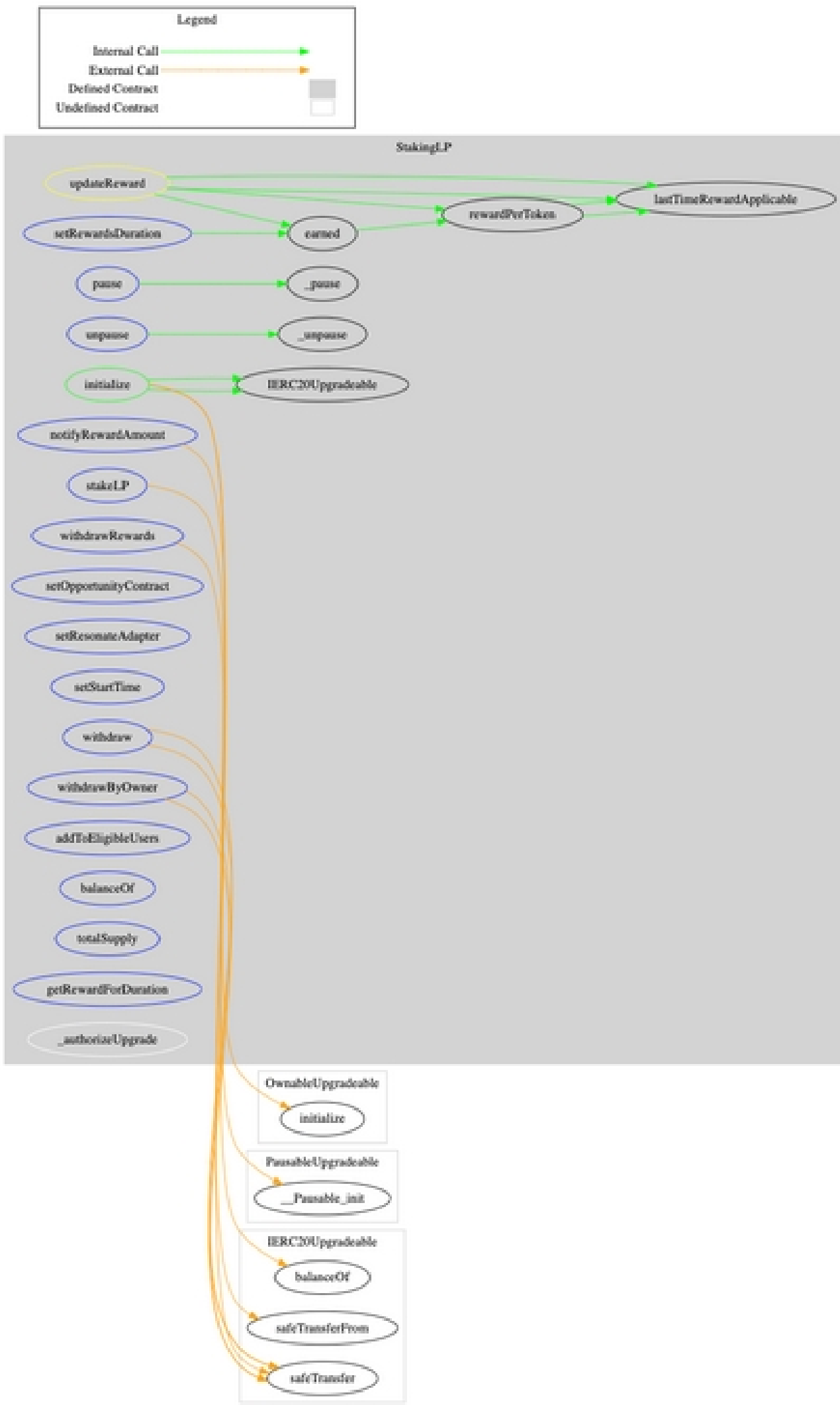
contracts/opportunity/Opportunity.sol



contracts/opportunity/PancakeOpportunity.sol



contracts/opportunity/StakingLP.sol



Notes: Above metrics and call graphs obtained from Solidity Metrics

Types of Severities

High

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level calls
- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Using block.timestamp
- Multiple Sends
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Testing

High Severity Issues

No issues found

Medium Severity Issues

No issues found

Low Severity Issues

No issues found

Informational Issues

1. Unnecessary checks

It appears some code checks allow 0 value and yet still check if uint is greater than or equal to zero. uint256 values will always be 0 or greater.

Opportunity.sol line 247, 252, 257, 262(_feePercentage >= 0,oe11)

Recommendation

Consider removing the above code snippets and any other that are similar to above but may not have been mentioned to save on gas costs.

Status

Resolved

2. Missing events

The missing event makes it difficult to track off-chain and impact off-chain monitoring and incident response functionality. An event should be emitted for critical operations, critical functions and significant transactions: It is especially important for admin functions and changes to emit events for transparency and monitoring so it is considered best practise to always have them logged. Consider the functions below:

BeefyMimaticUsdcOpportunity.sol functions lines 70 setSwapContract, 75 setRouter, and 80 setVault.

CrowdUsdtLpStakeOpportunity.sol functions lines 68 setSwapContract, 73 setRouter, and 78 setStakingLP

Opportunity.sol functions lines setFeeTo 241, setAddLiquidityFee 246, setRemoveLiquidityFee 251, setStakeFee 256, setUnstakeFee 261, setTokenA 266,setTokenB 271, setPair 276.

PancakeOpportunity.sol functions lines 109 setSwapContract, 114 setRouter, 119 setMasterChefV2, 124 setPId, 128 setRewardToken

Recommendation

Critical updates may need to emit events. We recommend emitting an event to log the update of the above functions or any other functions or operations deemed requiring event emission that may not be mentioned above.

Status

Resolved

Functional Testing

Some of the tests performed are mentioned below:

helpers/OwnableUpgradeable.sol

✓ initialize	function	PASS
✓ onlyOwner	modifier	PASS
✓ owner	function	PASS
✓ pendingOwner	function	PASS
✓ transferOwnership	function	PASS
✓ claimOwnership	function	PASS

libraries/UniERC20Upgradeable.sol

✓ isETH	function	PASS
✓ uniBalanceOf	function	PASS
✓ uniTransfer	function	PASS
✓ uniApprove	function	PASS

opportunity/BeefyMimaticUsdcOpportunity.sol

✓ initialize	function	PASS
✓ setSwapContract	function	PASS
✓ setRouter	function	PASS
✓ setVault	function	PASS
✓ swap	function	PASS
✓ addLiquidity	function	PASS
✓ removeLiquidity	function	PASS
✓ stake	function	PASS
✓ unstake	function	PASS

opportunity/CrowdUsdtLpOpportunity.sol

✓ initialize	function	PASS
✓ setSwapContract	function	PASS
✓ setRouter	function	PASS
✓ setStakingLp	function	PASS
✓ swap	function	PASS
✓ addLiquidity	function	PASS

✓ removeLiquidity	function	PASS
✓ stake	function	PASS
✓ unstake	function	PASS

opportunity/Opportunity.sol

✓ investByTokenATokenB	function	PASS
✓ investByTokenAOrTokenB	function	PASS
✓ investByToken	function	PASS
✓ investByLP	function	PASS
✓ leave	function	PASS
✓ setFeeTo	function	PASS
✓ setAddLiquidityFee	function	PASS
✓ setRemoveLiquidityFee	function	PASS
✓ setStakeFee	function	PASS
✓ setUnstakeFee	function	PASS
✓ setTokenA	function	PASS
✓ setTokenB	function	PASS
✓ setPair	function	PASS
✓ withdrawFunds	function	PASS
✓ pause	function	PASS
✓ unpause	function	PASS
✓ swap	function	PASS
✓ addLiquidity	function	PASS
✓ removeLiquidity	function	PASS
✓ stake	function	PASS
✓ unstake	function	PASS
✓ _initializeContracts	function	PASS
✓ _initializeFees	function	PASS
✓ _authorizeUpgrade	function	PASS
✓ _addLiquidity	function	PASS
✓ _removeLiquidity	function	PASS
✓ _stake	function	PASS
✓ _unstake	function	PASS
✓ _deductFee	function	PASS
✓ _calculateFee	function	PASS
✓ _transferFrom	function	PASS

opportunity/PancakeOpportunity.sol

✓ initialize	function	PASS
✓ updateRewards	modifier	PASS
✓ setSwapContract	function	PASS
✓ setRouter	function	PASS
✓ setMasterChefV2	function	PASS
✓ setPId	function	PASS
✓ setRewardToken	function	PASS
✓ getUserInfo	function	PASS
✓ withdrawRewards	function	PASS
✓ swap	function	PASS
✓ addLiquidity	function	PASS
✓ removeLiquidity	function	PASS
✓ stake	function	PASS
✓ unstake	function	PASS

libraries/UniERC20Upgradeable.sol

✓ initialize	function	PASS
✓ updateReward	modifier	PASS
✓ stakeLP	function	PASS
✓ withdrawRewards	function	PASS
✓ withdraw	function	PASS
✓ withdrawByOwner	function	PASS
✓ notifyRewardAmount	function	PASS
✓ setRewardsDuration	function	PASS
✓ setOpportunityContract	function	PASS
✓ setResonateAdapter	function	PASS
✓ setStartTime	function	PASS
✓ pause	function	PASS
✓ unpause	function	PASS
✓ addToEligibleUsers	function	PASS
✓ balanceOf	function	PASS
✓ totalSupply	function	PASS
✓ getRewardForDuration	function	PASS
✓ lastTimeRewardApplicable	function	PASS
✓ rewardPerToken	function	PASS
✓ earned	function	PASS

Automated Tests

No major issues were found. Some false positive errors were reported by the tools.
All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of the xEVMToken by SpaceFi. We performed our audit according to the procedure described above.

Some issues of Informational nature were found in this audit. Some suggestions and best practices are also provided in order to improve the code quality and security posture

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the Crowdswap Platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Crowdswap Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

700+
Audits Completed

$16B
Secured

700K
Lines of Code Audited

Follow Our Journey

Audit Report
February, 2023

For

CrowdSwap

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com