Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	ArcherSwap Protocol
Website:	archerswap.finance
Platform:	Core Chain
Language:	Solidity
Date:	February 20th, 2023

Table of contents

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 ArcherSwap team to perform the Security audit of the ArcherSwap Protocol 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 February 20th, 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

- ArcherSwap is a crypto world for users to trade, earn, and game. It's the best choice for projects on Core Chain with features including AMM, NFT, and GameFi.
- The ArcherSwap Contracts handle multiple contracts, and all contracts have different functions.
    - BowStakingToken: This contract handles swapping to and from xBOW, ArcherSwap's staking token. And the place where bow's live to create xbow.
    - SyrupBar: It is used for BOW staking.
    - LakeOfBow: LakeOfBow is MasterChef's left hand and kinda a wizard. He can brew Bow from pretty much anything! This contract handles "serving up" rewards for xBow holders by trading tokens collected from fees for Bow.
    - MasterChef: MasterChef is the master of BOW.
- The ArcherSwap Contracts have functions like adding a new pair and LPs, depositNFT, withdrawNFT, deposit, withdraw, reward, mint, swap, burn, skim, etc.

Audit scope

Name	Code Review and Security Analysis Report for ArcherSwap Protocol Smart Contracts
Platform	**Core Chain / Solidity**
File 1	MasterChef.sol
File 1 MD5 Hash	4E3505156A83EC77F419899CCBB51C9D
File 2	WETH9.sol
File 2 MD5 Hash	2FBAB491800E2F02C6D6B1970E6DE284
File 3	Oracle.sol
File 3 MD5 Hash	A72B18A4181306207A24212E4DB13244
File 4	SwapMining.sol
File 4 MD5 Hash	8DC6A01318201E3DEE26E16A55E27844
File 5	SyrupBar.sol
File 5 MD5 Hash	C7CBC8D1FF1B97D83A53F44280CC8622
File 6	ArcherswapFactory.sol
File 6 MD5 Hash	A35017EA5C8EB9DAB1D47579FF10CDF8
File 7	BowToken.sol
File 7 MD5 Hash	CF6CA2CF455597E89FAC72FFB3B4C63C
File 8	Router.sol
File 8 MD5 Hash	53940C5EBBAC717837DB747DAE355209
File 9	BowStakingToken.sol
File 9 MD5 Hash	0F1172ACC33458662B577156776C796D
File 10	Multicall.sol
File 10 MD5 Hash	B22CA4A854478127BCB7BF23881EB4E6
File 11	LakeOfBow.sol
File 11 MD5 Hash	86294C6B2E61505AF76B8DDA8C92E7AA

File 12	NFTController.sol
File 12 MD5 Hash	6AAE550160948A4C6E4028309D9CC9DA
File 13	Pair.sol
File 13 MD5 Hash	FC98D007A39E81DB71A49D0BEFDB725A
Audit Date	February 20th,2023

Email: audit@EtherAuthority.io

Claimed Smart Contract Features

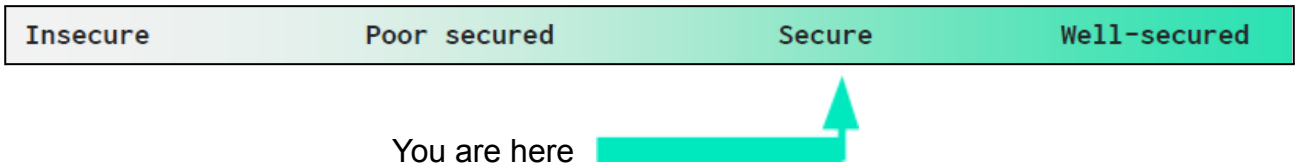
Claimed Feature Detail	Our Observation
File 1 MasterChef.sol • NFT Boost Rate: 1%. • MasterChef is the master of BOW. • Maximum Cake per Sec: 10 Quintillion. **Ownership Control:** • Owner can add a new lp to the pool. • Owner can update the given pool's BOW allocation point. • Owner can update the cake token reward per second, with a cap of max cake per second. • Owner can set the Nft boost rate range. • Owner can update the trade mining contract address. • Owner can update the reserve address by the previous reserve address.	**YES, This is valid.**
File 2 Oracle.sol • Oracle can update token addresses.	**YES, This is valid.**
File 3 SwapMining.sol • Owner can add a new pair. • Owner can update the allocPoint of the pool. • Owner can set a halving period value. • Owner can swap Mining.	**YES, This is valid.**
File 4 SyrupBar.sol • Name: ArcherSwapBar Token • Symbol: SYRUP • SyrupBar used for BOW staking.	**YES, This is valid.**

Ownership Control: • Owner can create a `_amount` token to `_to` by the MasterChef owner. • Owner can burn an amount from the address.	
File 5 ArcherswapFactory.sol • Owner can set a fee address.	**YES, This is valid.**
File 6 BowStakingToken.sol • Name: Bow Staking Token • Symbol: xBOW • Decimals: 18 **Other Specifications:** • xBOW is the place where bow's live to create xbows. • xBOW contract handles swapping to and from xBOW, ArcherSwap's staking token.	**YES, This is valid.**
File 7 BowToken.sol • Name: ArcherSwap Token • Symbol: BOW • Decimals: 18	**YES, This is valid.**
File 8 LakeOfBow.sol • LakeOfBow is MasterChef's left hand and kinda a wizard. He can brew Bow from pretty much anything! • This contract handles "serving up" rewards for xBow holders by trading tokens collected from fees for Bow. **Ownership Control:** • Owner can set anyAuth to true and allows anyone to call functions protected by onlyAuth. • Owner can set the bridge address.	**YES, This is valid.**

File 9 Multicall.sol • Multicall contract has aggregate results from multiple read-only function calls.	**YES, This is valid.**
File 10 ArcherswapRouter.sol • Owner can set a swap mining address.	**YES, This is valid.**
File 11 WETH9.sol • Decimals: 18 • Weth9 has withdrawal amount, deposit amount.	**YES, This is valid.**
File 12 NFTController.sol • Owner can set a whitelist address. • Owner can set the default Boost Rate 1%.	**YES, This is valid.**
File 13 ArcherswapPair.sol • Owner can be called once by the factory at time of deployment.	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. These contracts do 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, 0 medium and 1 low and some 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	Moderated
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Moderated
	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	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 13 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in the ArcherSwap Protocol 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 ArcherSwap Protocol.

The ArcherSwap Protocol 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 **not** well commented on smart contracts.

Documentation

We were given an ArcherSwap Protocol smart contract code in the form of https://scan.coredao.org weblink. The hash of that code is mentioned above in the table.

As mentioned above, code parts are **not well** commented. But 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 website: https://archerswap.finance 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

MasterChef.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	getBoost	read	Passed	No Issue
7	getSlots	read	Passed	No Issue
8	getTokenIds	read	Passed	No Issue
9	updateMultiplier	write	access only Owner	No Issue
10	poolLength	external	Passed	No Issue
11	add	write	access only Owner	No Issue
12	set	write	access only Owner	No Issue
13	depositNFT	write	Passed	No Issue
14	withdrawNFT	write	Passed	No Issue
15	getMultiplier	read	Passed	No Issue
16	pendingCake	external	Passed	No Issue
17	massUpdatePools	write	Passed	No Issue
18	updatePool	write	Passed	No Issue
19	deposit	write	Passed	No Issue
20	withdraw	write	Passed	No Issue
21	emergencyWithdraw	write	Passed	No Issue
22	safeCakeTransfer	internal	Passed	No Issue
23	setCakePerSecond	external	access only Owner	No Issue
24	setNftController	write	access only Owner	No Issue
25	setNftBoostRate	write	access only Owner	No Issue
26	setDevaddr	write	Passed	No Issue
27	setReserveaddr	write	Passed	No Issue
28	setMiningaddr	external	access only Owner	No Issue

NFTController.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	mint	write	Passed	No Issue
3	owner	read	Passed	No Issue
4	onlyOwner	modifier	Passed	No Issue
5	renounceOwnership	write	access only Owner	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	getBoostRate	read	Passed	No Issue

8	setWhitelist	external	access only Owner	No Issue
9	setDefaultBoostRate	external	access only Owner	No Issue
10	setBoostRate	external	access only Owner	No Issue
11	mint	write	access only Owner	No Issue

Pair.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	lock	modifier	Passed	No Issue
3	getReserves	read	Passed	No Issue
4	_safeTransfer	write	Passed	No Issue
5	initialize	external	Passed	No Issue
6	_update	write	Passed	No Issue
7	_mintFee	write	Passed	No Issue
8	mint	external	Passed	No Issue
9	burn	external	Passed	No Issue
10	swap	external	Passed	No Issue
11	skim	external	Passed	No Issue
12	sync	external	Passed	No Issue

SwapMining.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	poolLength	read	Passed	No Issue
7	addPair	write	Critical operation lacks event log	Refer Audit Findings
8	setPair	write	Critical operation lacks event log	Refer Audit Findings
9	setArcherswapPerSecond	write	access only Owner	No Issue
10	addWhitelist	write	access only Owner	No Issue
11	delWhitelist	write	access only Owner	No Issue
12	getWhitelistLength	read	Passed	No Issue
13	isWhitelist	read	Passed	No Issue
14	getWhitelist	read	Passed	No Issue
15	setHalvingPeriod	write	access only Owner	No Issue
16	setRouter	write	access only Owner	No Issue
17	setOracle	write	access only Owner	No Issue
18	phase	read	Passed	No Issue

19	phase	read	Passed	No Issue
20	reward	read	Passed	No Issue
21	reward	read	Passed	No Issue
22	getBowReward	read	Passed	No Issue
23	massMintPools	write	Passed	No Issue
24	mint	write	Critical operation lacks event log	Refer Audit Findings
25	onlyRouter	modifier	Passed	No Issue
26	swap	write	access only Router	No Issue
27	getQuantity	read	Passed	No Issue
28	takerWithdraw	write	Critical operation lacks event log	Refer Audit Findings
29	getUserReward	read	Passed	No Issue
30	getTotalUserReward	read	Passed	No Issue
31	getPoolInfo	read	Passed	No Issue
32	ownerWithdraw	write	Critical operation lacks event log	Refer Audit Findings
33	addBlacklist	external	access only Owner	No Issue
34	removeBlacklist	external	access only Owner	No Issue
35	safeBowTransfer	internal	Passed	No Issue

SyrupBar.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getOwner	external	Passed	No Issue
3	name	read	Passed	No Issue
4	decimals	read	Passed	No Issue
5	symbol	read	Passed	No Issue
6	totalSupply	read	Passed	No Issue
7	balanceOf	read	Passed	No Issue
8	transfer	write	Passed	No Issue
9	allowance	write	Passed	No Issue
10	approve	write	Passed	No Issue
11	transferFrom	write	Passed	No Issue
12	increaseAllowance	write	Passed	No Issue
13	decreaseAllowance	write	Passed	No Issue
14	mint	write	access only Owner	No Issue
15	_transfer	internal	Passed	No Issue
16	_mint	internal	Passed	No Issue
17	_burn	internal	Passed	No Issue
18	_approve	internal	Passed	No Issue
19	_burnFrom	internal	Passed	No Issue
20	mint	write	access only Owner	No Issue
21	burn	write	access only Owner	No Issue
22	safeCakeTransfer	write	access only Owner	No Issue

23	delegates	external	Passed	No Issue
24	delegate	external	Passed	No Issue
25	getCurrentVotes	external	Passed	No Issue
26	delegateBySig	external	Passed	No Issue
27	getPriorVotes	external	Passed	No Issue
28	_delegate	internal	Passed	No Issue
29	_moveDelegates	internal	Passed	No Issue
30	_writeCheckpoint	internal	Passed	No Issue
31	safe32	internal	Passed	No Issue
32	getChainId	internal	Passed	No Issue

WETH9.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	WETH9	write	Passed	No Issue
3	deposit	write	Passed	No Issue
4	withdraw	write	Passed	No Issue
5	totalSupply	read	Passed	No Issue
6	approve	write	Passed	No Issue
7	transfer	write	Passed	No Issue
8	transferFrom	write	Passed	No Issue

Oracle.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	update	external	Passed	No Issue
3	computeAmountOut	write	Passed	No Issue
4	consult	external	Passed	No Issue

ArcherswapFactory.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	allPairsLength	external	Passed	No Issue
3	expectPairFor	read	Passed	No Issue
4	createPair	external	Passed	No Issue
5	setFeeTo	external	Passed	No Issue
6	setFeeToSetter	external	Passed	No Issue

ArcherswapRouter.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	ensure	modifier	Passed	No Issue
3	setSwapMining	write	ow	No Issue
4	receive	external	Passed	No Issue
5	_addLiquidity	internal	Passed	No Issue
6	addLiquidity	external	Passed	No Issue
7	addLiquidityETH	external	Passed	No Issue
8	removeLiquidity	write	Passed	No Issue
9	removeLiquidityETH	write	Passed	No Issue
10	removeLiquidityWithPermit	external	Passed	No Issue
11	removeLiquidityETHWithPermit	external	Passed	No Issue
12	removeLiquidityETHSupportingFeeOnTransferTokens	write	Passed	No Issue
13	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	external	Passed	No Issue
14	_swap	internal	Passed	No Issue
15	swapExactTokensForTokens	external	Passed	No Issue
16	swapTokensForExactTokens	external	Passed	No Issue
17	swapExactETHForTokens	external	Passed	No Issue
18	swapTokensForExactETH	external	Passed	No Issue
19	swapExactTokensForETH	external	Passed	No Issue
20	swapETHForExactTokens	external	Passed	No Issue
21	_swapSupportingFeeOnTransferTokens	internal	Passed	No Issue
22	swapExactTokensForTokensSupportingFeeOnTransferTokens	external	Passed	No Issue
23	swapExactETHForTokensSupportingFeeOnTransferTokens	external	Passed	No Issue
24	swapExactTokensForETHSupportingFeeOnTransferTokens	external	Passed	No Issue
25	quote	write	Passed	No Issue
26	getAmountOut	write	Passed	No Issue
27	getAmountIn	write	Passed	No Issue
28	getAmountsOut	read	Passed	No Issue
29	getAmountsIn	read	Passed	No Issue

BowToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	mintFor	write	access only Owner	No Issue
3	mint	write	access only Owner	No Issue

4	delegates	external	Passed	No Issue
5	delegate	external	Passed	No Issue
6	delegateBySig	external	Passed	No Issue
7	getCurrentVotes	external	Passed	No Issue
8	getPriorVotes	external	Passed	No Issue
9	_delegate	internal	Passed	No Issue
10	_moveDelegates	internal	Passed	No Issue
11	_writeCheckpoint	internal	Passed	No Issue
12	safe32	internal	Passed	No Issue
13	getChainId	internal	Passed	No Issue

BowStakingToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getOwner	external	Passed	No Issue
3	name	read	Passed	No Issue
4	decimals	read	Passed	No Issue
5	symbol	read	Passed	No Issue
6	totalSupply	read	Passed	No Issue
7	balanceOf	read	Passed	No Issue
8	transfer	write	Passed	No Issue
9	allowance	write	Passed	No Issue
10	approve	write	Passed	No Issue
11	transferFrom	write	Passed	No Issue
12	increaseAllowance	write	Passed	No Issue
13	decreaseAllowance	write	Passed	No Issue
14	mint	write	access only Owner	No Issue
15	_transfer	internal	Passed	No Issue
16	_mint	internal	Passed	No Issue
17	_burn	internal	Passed	No Issue
18	_approve	internal	Passed	No Issue
19	_burnFrom	internal	Passed	No Issue
20	stakedTime	read	Passed	No Issue
21	canWithdraw	read	Passed	No Issue
22	setDelayToWithdraw	external	Passed	No Issue
23	enter	write	Critical operation lacks event log	Refer Audit Findings
24	leave	write	Critical operation lacks event log	Refer Audit Findings
25	BOWBalance	external	Passed	No Issue
26	xBOWForBOW	external	Passed	No Issue
27	BOWForxBOW	external	Passed	No Issue
28	burn	write	Passed	No Issue
29	mint	write	Passed	No Issue
30	transferFrom	write	Passed	No Issue

31	transfer	write	Passed	No Issue
32	_initDelegates	internal	Passed	No Issue
33	delegates	external	Passed	No Issue
34	delegate	external	Passed	No Issue
35	delegateBySig	external	Passed	No Issue
36	getCurrentVotes	external	Passed	No Issue
37	getPriorVotes	external	Passed	No Issue
38	_delegate	internal	Passed	No Issue
39	_moveDelegates	internal	Passed	No Issue
40	_writeCheckpoint	internal	Passed	No Issue
41	safe32	internal	Passed	No Issue
42	getChainId	internal	Passed	No Issue
43	setAdmin	write	Passed	No Issue

LakeOfBow.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	onlyAuth	modifier	Passed	No Issue
7	addAuth	external	access only Owner	No Issue
8	revokeAuth	external	access only Owner	No Issue
9	setAnyAuth	external	access only Owner	No Issue
10	setBridge	external	access only Owner	No Issue
11	setDevCut	external	access only Owner	No Issue
12	setDevAddr	external	access only Owner	No Issue
13	bridgeFor	read	Passed	No Issue
14	onlyEOA	modifier	Passed	No Issue
15	convert	external	access only Auth	No Issue
16	convertMultiple	external	access only Auth	No Issue
17	_convert	internal	Passed	No Issue
18	_convertStep	internal	Passed	No Issue
19	_swap	internal	Passed	No Issue
20	_toBOW	internal	Passed	No Issue
21	getAmountOut	internal	Passed	No Issue

Multicall.sol

Functions

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

3	getEthBalance	read	Passed	No Issue
4	getBlockHash	read	Passed	No Issue
5	getLastBlockHash	read	Passed	No Issue
6	getCurrentBlockTimestamp	read	Passed	No Issue
7	getCurrentBlockDifficulty	read	Passed	No Issue
8	getCurrentBlockGasLimit	read	Passed	No Issue
9	getCurrentBlockCoinbase	read	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.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Critical operation lacks event log:

Missing event log for:

MasterChef.sol

- add
- set
- updatePool

BowStakingToken.sol

- enter.
- leave

SwapMining.sol

- addPair
- setPair
- mint
- ownerWithdraw
- takerWithdraw

Resolution: Write an event log for listed events.

Very Low / Informational / Best practices:

(1) Use the latest solidity version: **- BowToken.sol, MockToken.sol, Syrupbar.sol, BowStakingToken.sol, WETH9.sol**

Using the latest solidity will prevent any compiler-level bugs.

.

Resolution: We suggest using the latest solidity version.

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:

MasterChef.sol

- updateMultiplier: Owner can update multiplier number value.
- add: Owner can add a new lp to the pool.
- set: Owner can update the given pool's BOW allocation point.
- setCakePerSecond: Owner can update cake token reward per second, with a cap of max cake per second.
- setNftController: Owner can set Nft controller address.
- setNftBoostRate: Owner can set Nft boost rate range.
- setMiningaddr: Owner can update trade mining contract address.
- setDevaddr: Owner can update dev address by the previous dev address.
- setReserveaddr: Owner can update reserve address by the previous reserve address.

NFTController.sol

- setWhitelist: Owner can set whitelist address.
- setDefaultBoostRate: Owner can set default Boost Rate 1%.

- setBoostRate: Owner can set default Boost Rate 1%.

SyrupBar.sol

- mint: Owner can create `_amount` token to `_to` by MasterChef owner.
- burn: Owner can burn an amount from the address.
- safeCakeTransfer: Owner can save cake transfer function, just in case if rounding error causes pool to not have enough Bows.

SwapMining.sol

- addPair: Owner can add new pair.
- setPair: Owner can update the allocPoint of the pool.
- setArcherswapPerSecond: Owner can set the number of bow produced by each second.
- addWhitelist: Owner can add new wallet address in whitelist.
- delWhitelist: Owner can remove wallet address from the whitelist.
- setHalvingPeriod: Owner can set halving period value.
- setRouter: Owner can set new router address.
- setOracle: Owner can set new oracle address.
- ownerWithdraw: Owner can withdraw amount from wallet address.
- addBlacklist: Owner can add wallet address in blacklist.
- removeBlacklist: Owner can remove wallet address from the blacklist.
- swap: Owner can swap Mining.

BowToken.sol

- mintFor: Owner can create `_amount` token to `_to` by masterchef owner.
- mint: Owner can mint value from owner wallet.

LakeOfBow.sol

- addAuth: Owner can add a new auth wallet address.
- revokeAuth: Owner can remove auth wallet address.
- setAnyAuth: Owner can set anyAuth to true and allows anyone to call functions protected by onlyAuth.
- setBridge: Owner can set bridge address.
- setDevCut: Owner can set dev cut amount.

- setDevAddr: Owner can set dev address.
- convert: Auth can convert token value.
- convertMultiple: Auth can convert multiple token values.

ArcherswapFactory.sol

- setFeeTo: Owner can set fee address.
- setFeeToSetter: Owner can set fee setter address.

ArcherswapPair.sol

- initialize: Owner can be called once by the factory at time of deployment.

ArcherswapRouter.sol

- setSwapMining: Owner can set swap mining address.

BowStakingToken.sol

- setAdmin: Owner can update admin address by the previous admin.

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 https://scan.coredao.org weblink. And we have used all possible tests based on given objects as files. We have not observed any major issues in the smart contracts. So, **it's good to go to production**.

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.

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

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.

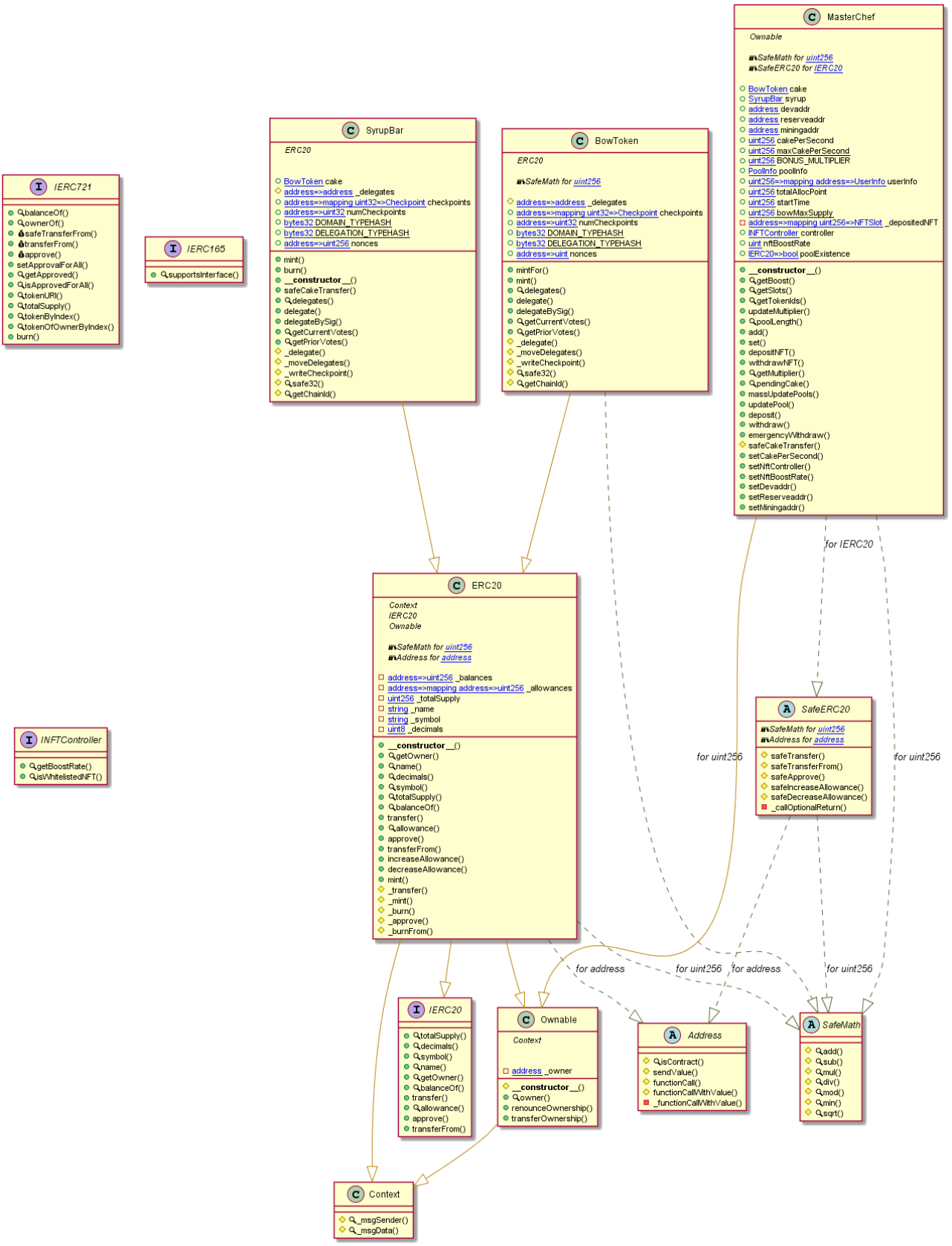
Technical Disclaimer

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.

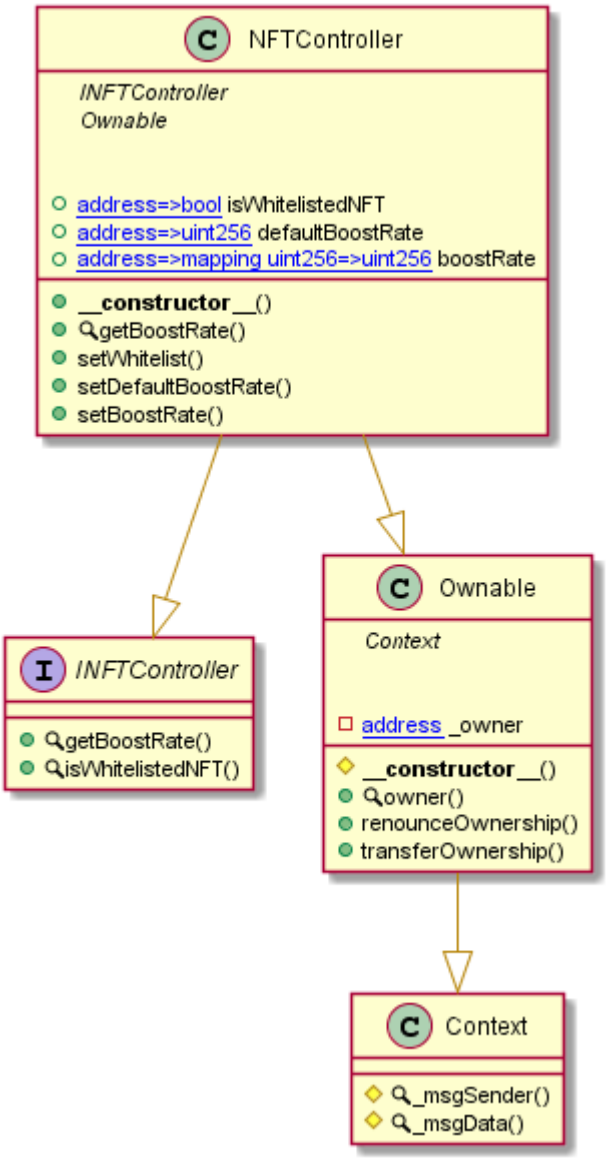
Appendix

Code Flow Diagram - ArcherSwap Protocol

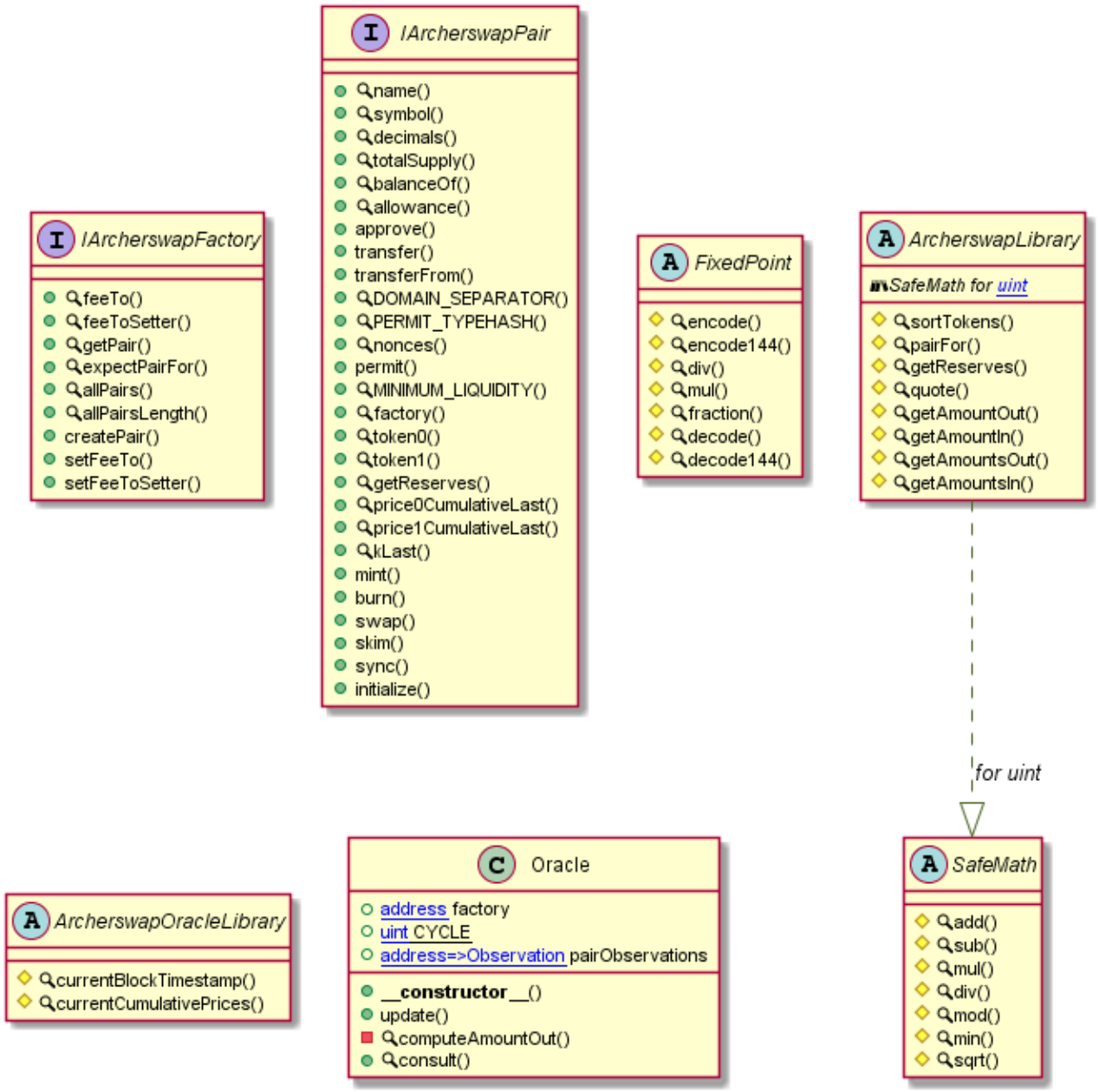
MasterChef Diagram



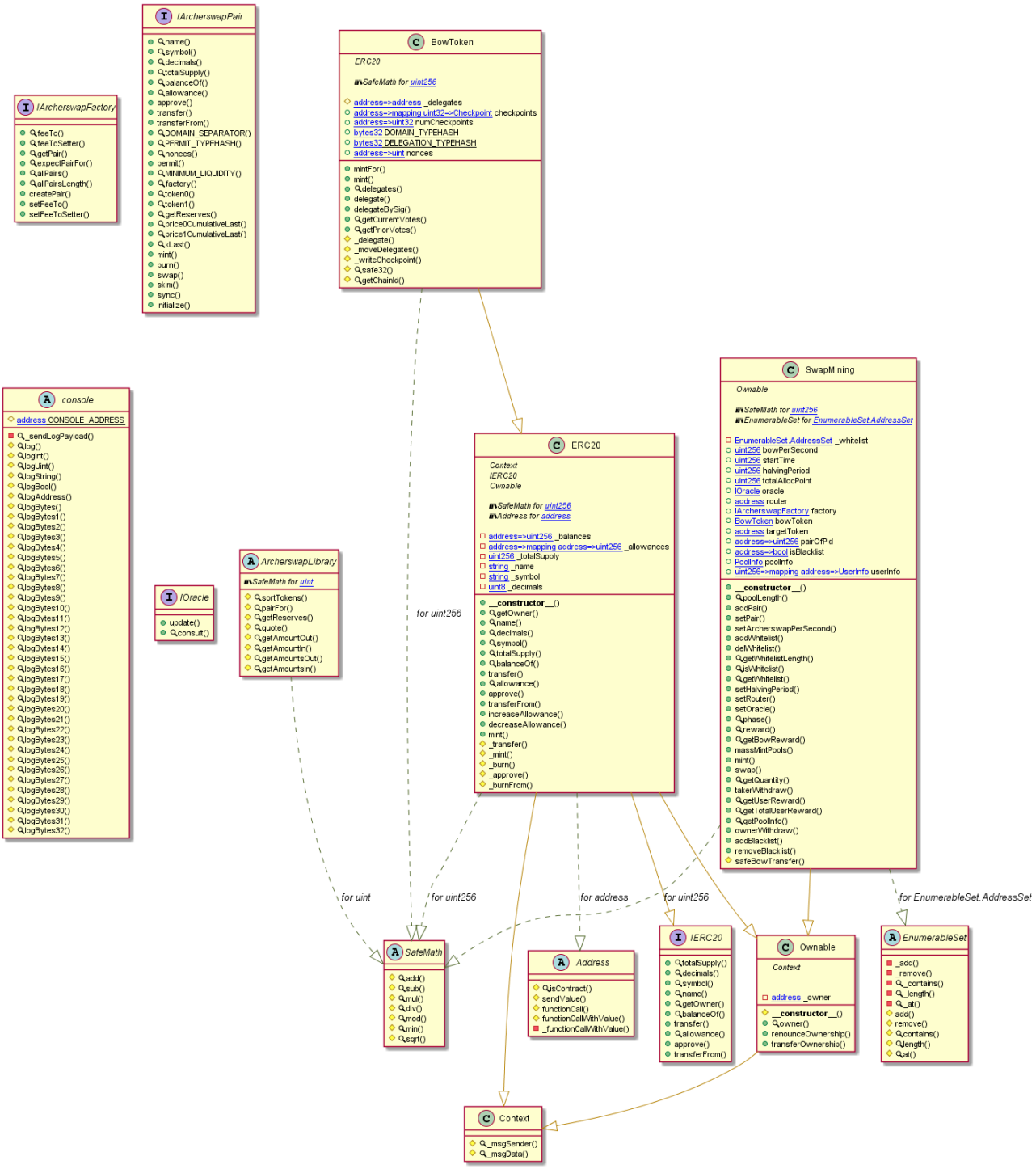
NFTController Diagram



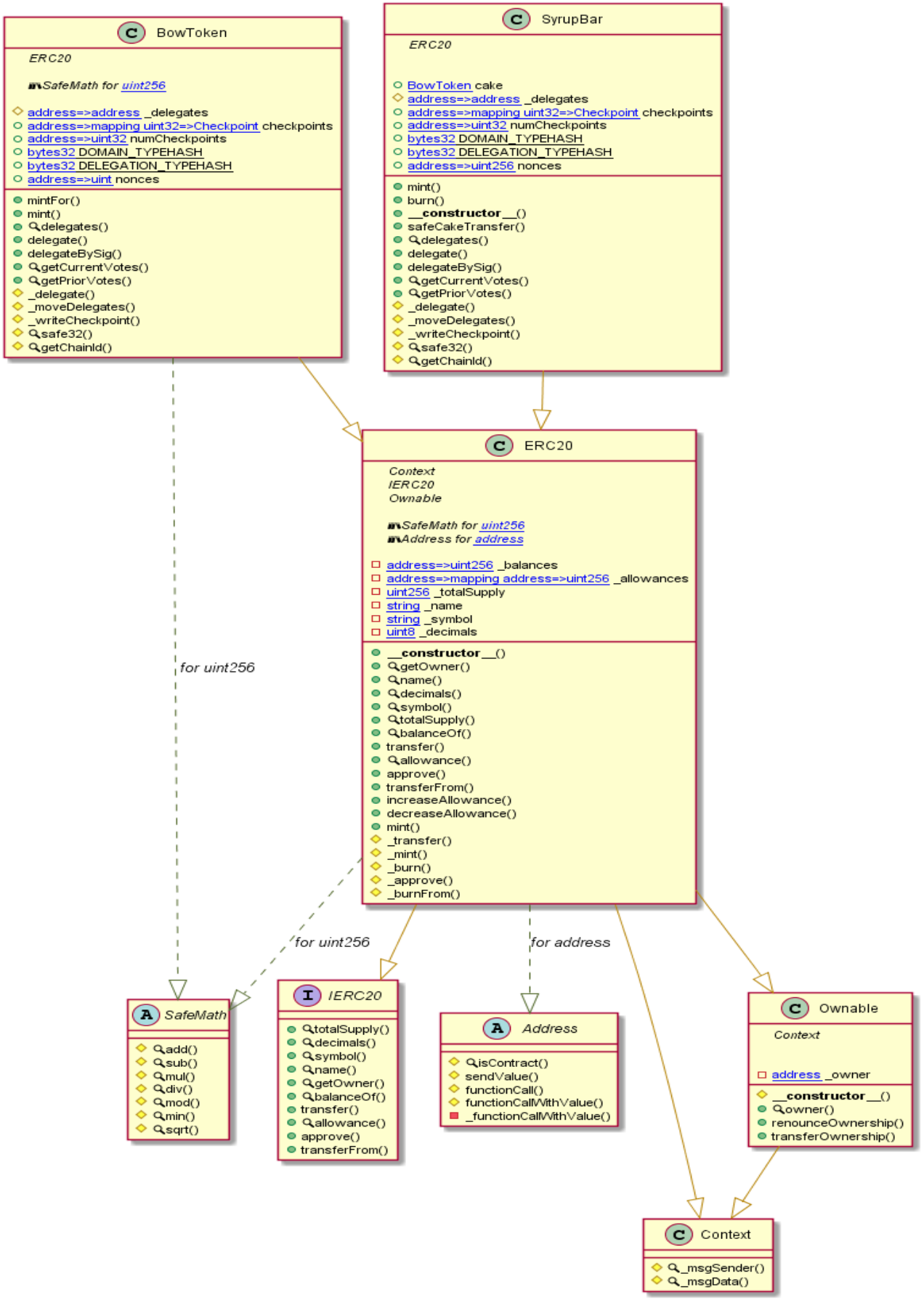
Oracle Diagram



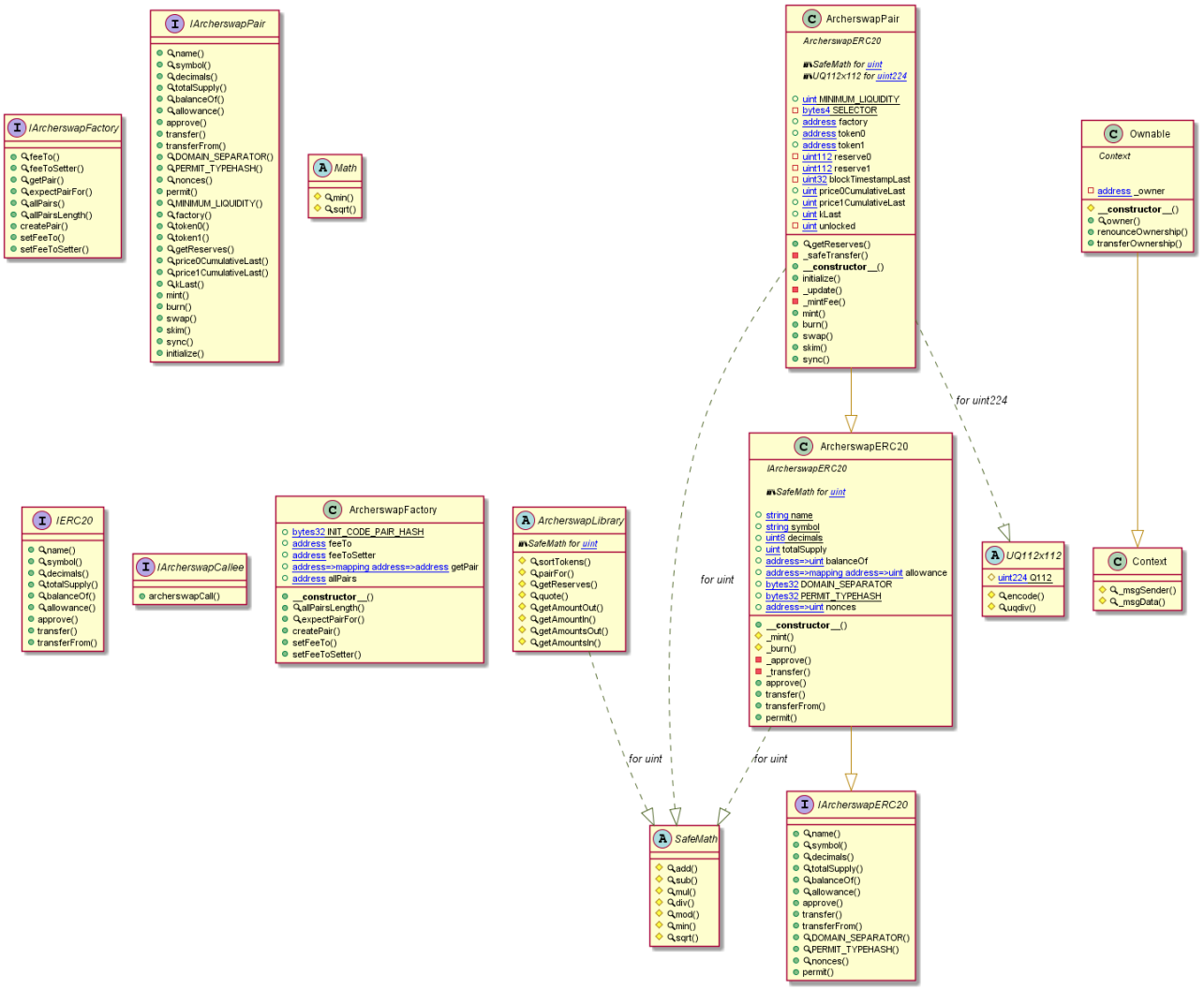
SwapMining Diagram



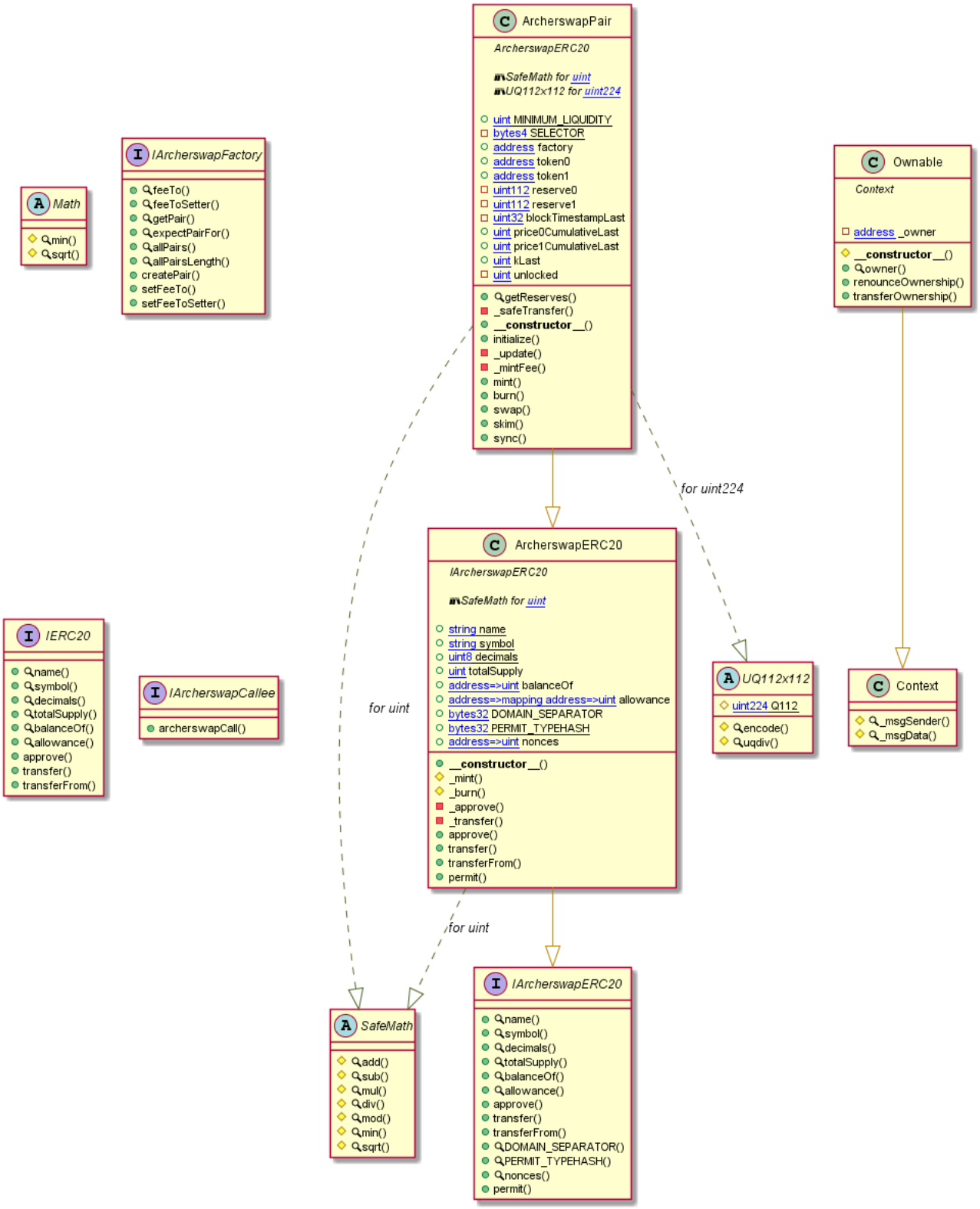
SyrupBar Diagram



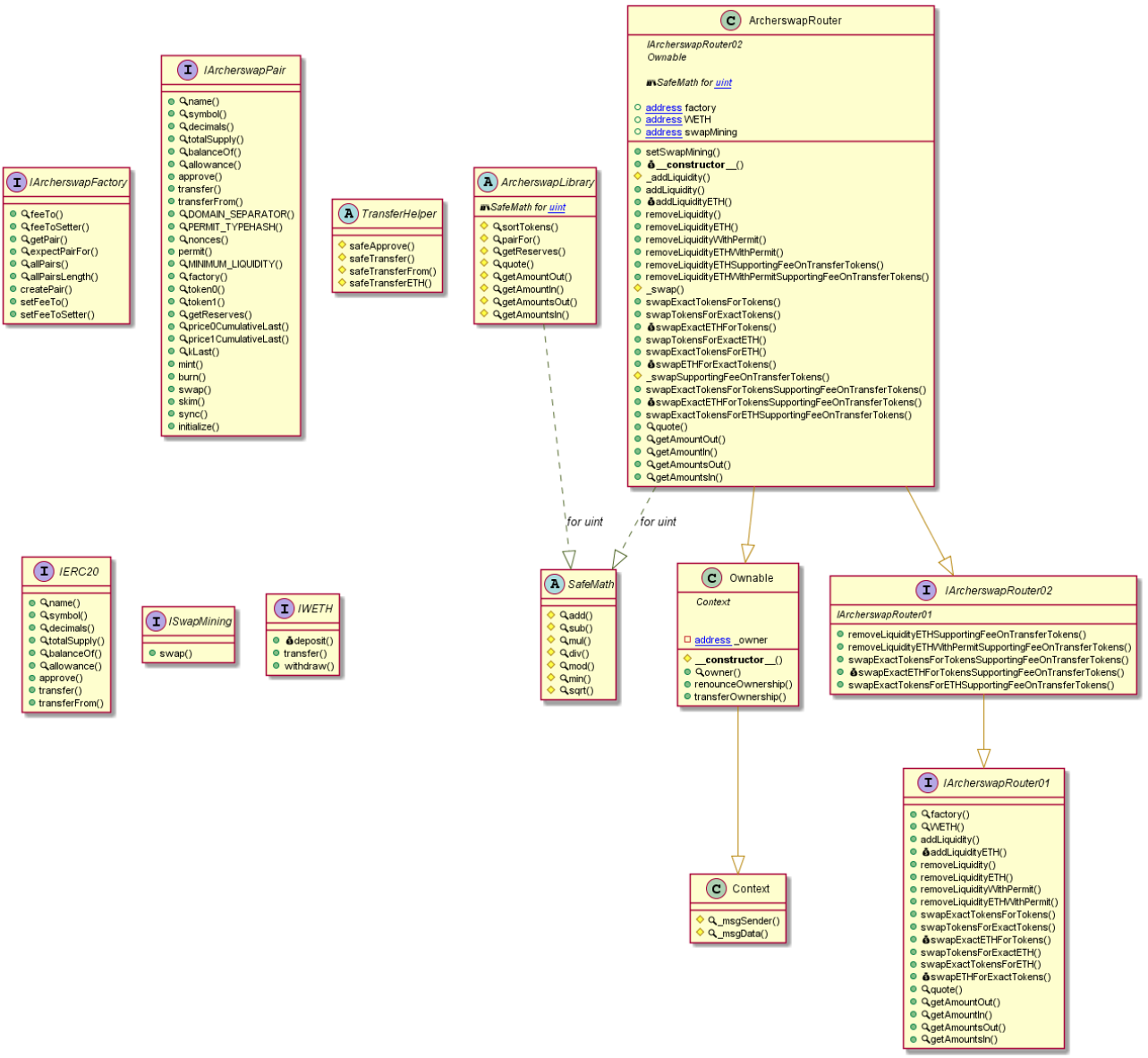
ArcherswapFactory Diagram



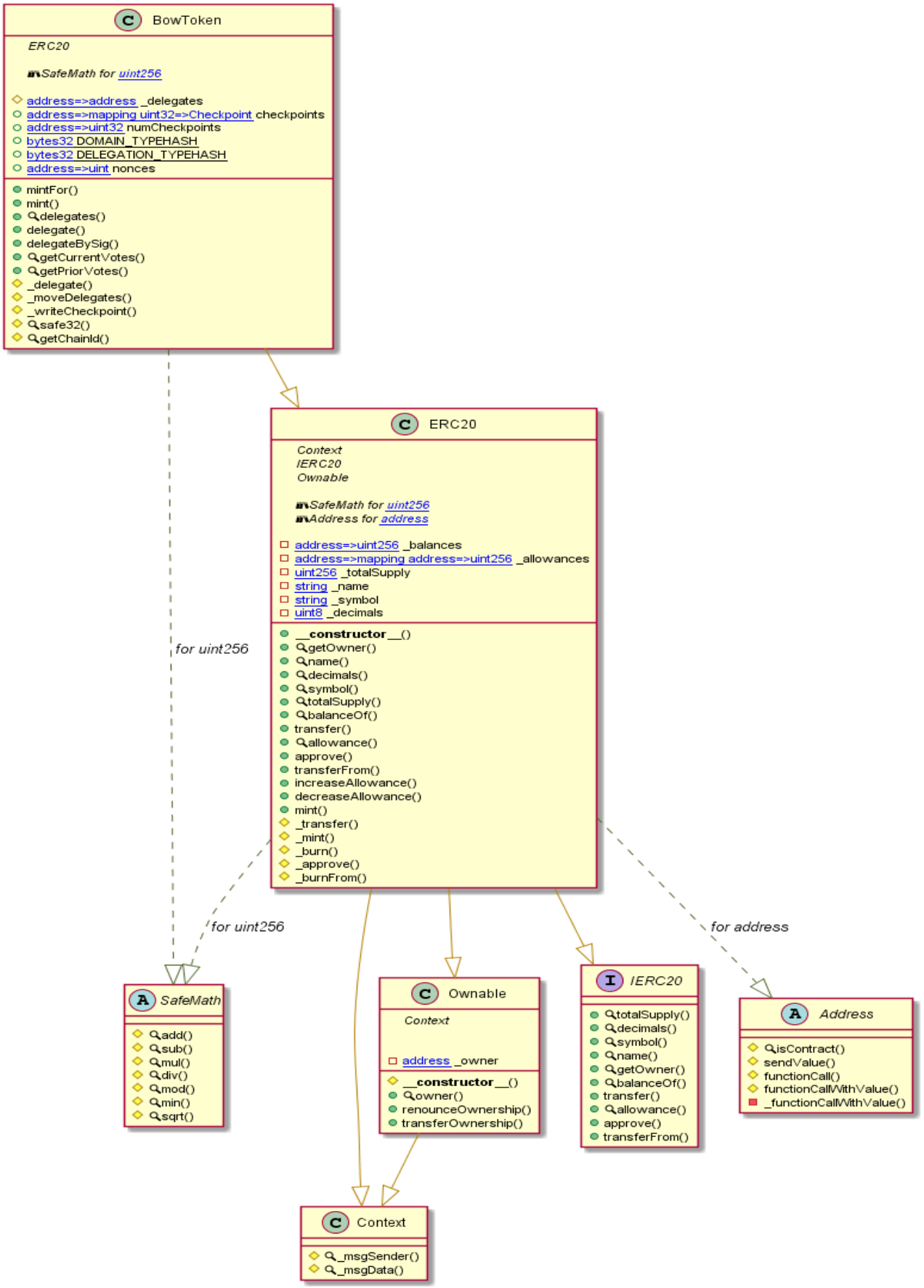
ArcherswapPair Diagram



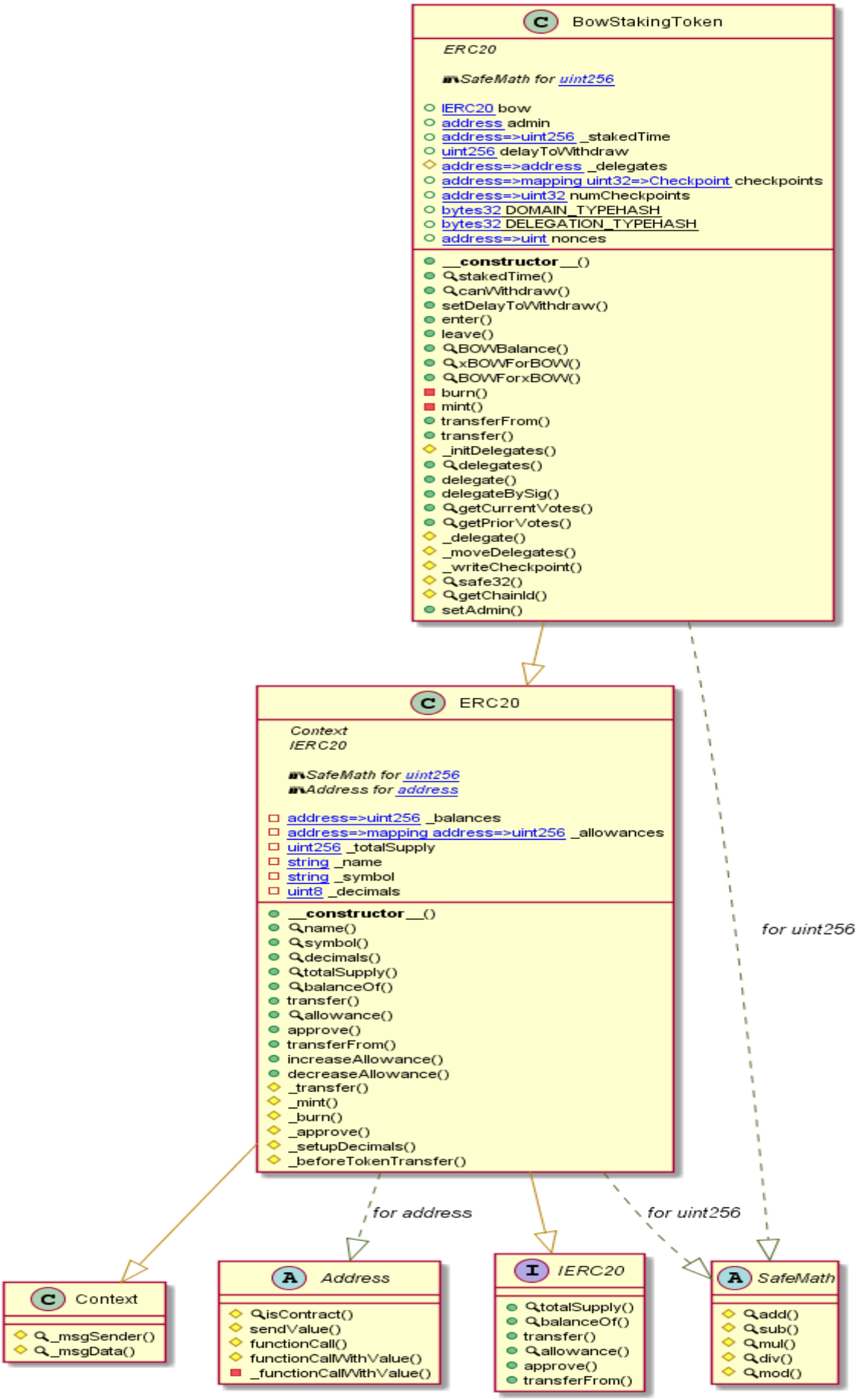
ArcherswapRouter Diagram



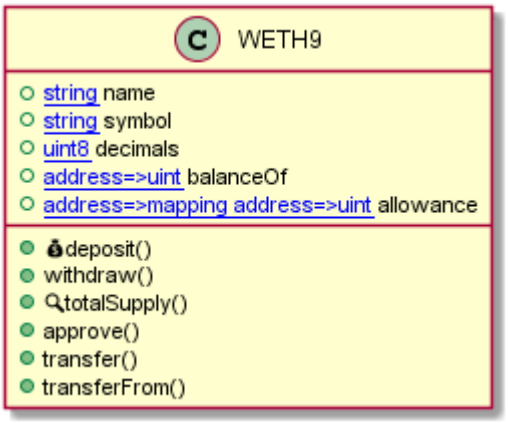
BowToken Diagram



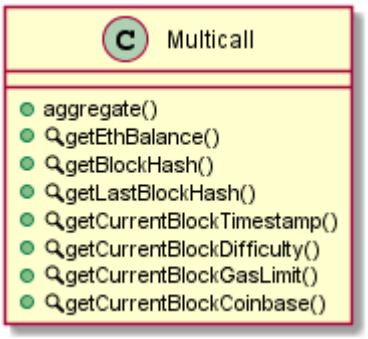
BowStakingToken Diagram



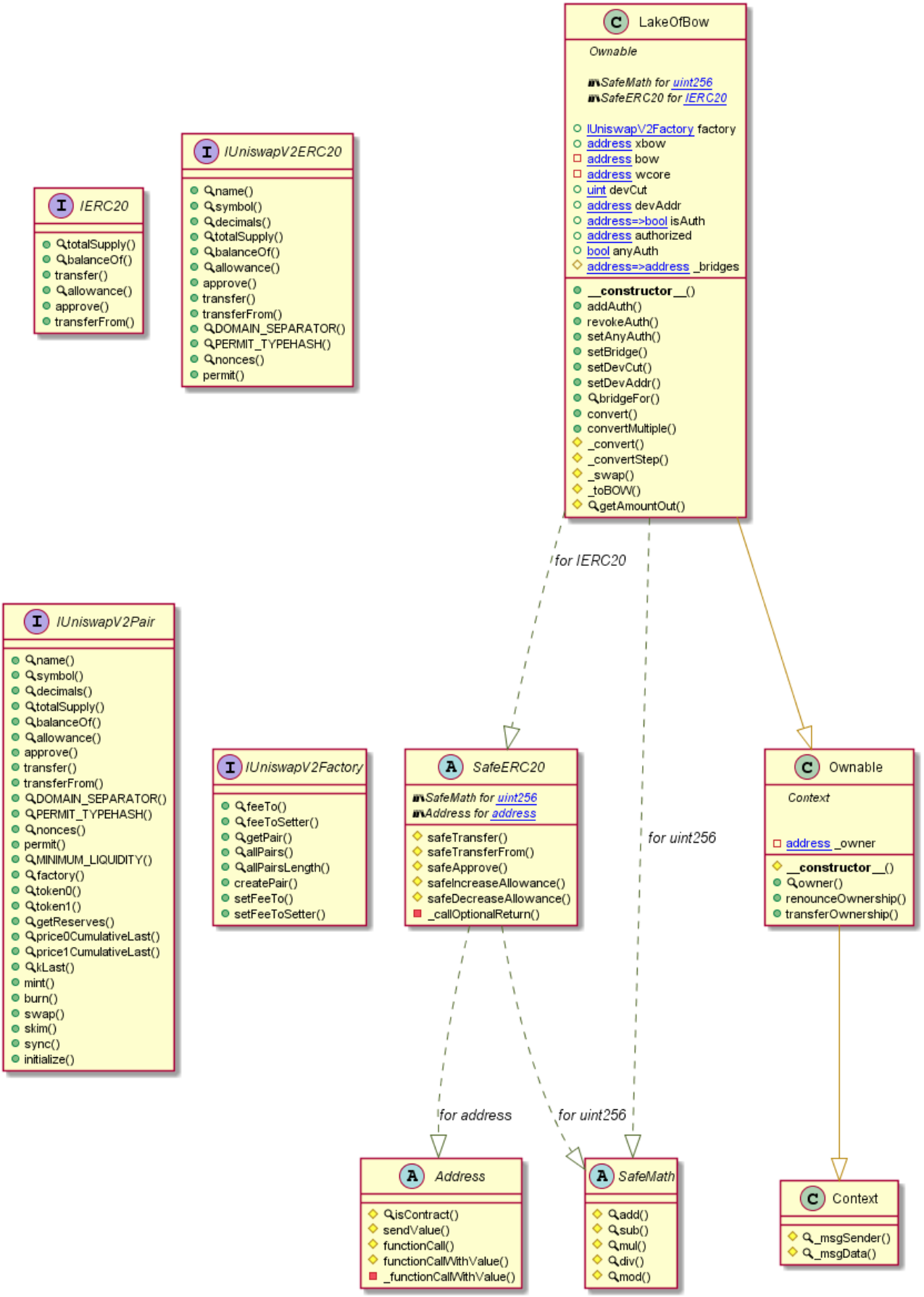
WETH9 Diagram



Multicall Diagram



LakeOfBow Diagram



Slither Results Log

Slither log >> MasterChef.sol

```
Address.isContract(address) (MasterChef.sol#30-41) uses assembly
        - INLINE ASM (MasterChef.sol#37-39)
Address._functionCallWithValue(address,bytes,uint256,string) (MasterChef.sol#138-164) uses assembly
        - INLINE ASM (MasterChef.sol#156-159)
BowToken.getChainId() (MasterChef.sol#1015-1019) uses assembly
        - INLINE ASM (MasterChef.sol#1017)
SyrupBar.getChainId() (MasterChef.sol#1295-1301) uses assembly
        - INLINE ASM (MasterChef.sol#1297-1299)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

MasterChef.nonDuplicatedLP(IERC20) (MasterChef.sol#1446-1449) compares to a boolean constant:
        -require(bool,string)(poolExistence[_lpToken] == false,nonDuplicated: Duplicated LPToken) (MasterChef.sol#1447)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```

```
Redundant expression "this (MasterChef.sol#407)" inContext (MasterChef.sol#401-410)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

MasterChef.cake (MasterChef.sol#1349) should be immutable
MasterChef.startTime (MasterChef.sol#1376) should be immutable
MasterChef.syrup (MasterChef.sol#1351) should be immutable
SyrupBar.cake (MasterChef.sol#1037) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
MasterChef.sol analyzed (13 contracts with 84 detectors), 123 result(s) found
```

Slither log >> NFTController.sol

```
Context._msgData() (NFTController.sol#16-19) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Redundant expression "this (NFTController.sol#17)" inContext (NFTController.sol#11-20)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
NFTController.sol analyzed (4 contracts with 84 detectors), 2 result(s) found
```

Slither log >> Oracle.sol

```
Oracle.constructor(address).factory_ (Oracle.sol#429) lacks a zero-check on :
                - factory = factory_ (Oracle.sol#430)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

ArcherswapOracleLibrary.currentCumulativePrices(address) (Oracle.sol#392-410) uses timestamp for comparisons
        Dangerous comparisons:
        - blockTimestampLast != blockTimestamp (Oracle.sol#401)
Oracle.update(address,address) (Oracle.sol#434-447) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(timeElapsed >= CYCLE,MDEXOracle: PERIOD_NOT_ELAPSED) (Oracle.sol#442)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
```

```
Pragma version>=0.6.6 (Oracle.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function IArcherswapPair.DOMAIN_SEPARATOR() (Oracle.sol#38) is not in mixedCase
Function IArcherswapPair.PERMIT_TYPEHASH() (Oracle.sol#39) is not in mixedCase
Function IArcherswapPair.MINIMUM_LIQUIDITY() (Oracle.sol#56) is not in mixedCase
Struct FixedPoint.uq112x112 (Oracle.sol#329-331) is not in CapWords
Struct FixedPoint.uq144x112 (Oracle.sol#335-337) is not in CapWords
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable ArcherswapOracleLibrary.currentCumulativePrices(address).price0Cumulative (Oracle.sol#394) is too similar to Archerswa
pOracleLibrary.currentCumulativePrices(address).price1Cumulative (Oracle.sol#394)
Variable Oracle.update(address,address).price0Cumulative (Oracle.sol#443) is too similar to Oracle.update(address,address).pric
e1Cumulative (Oracle.sol#443)
Variable Oracle.consult(address,uint256,address).price0Cumulative (Oracle.sol#466) is too similar to Oracle.consult(address,uin
t256,address).price1Cumulative (Oracle.sol#466)
Variable Oracle.update(address,address).price0Cumulative (Oracle.sol#443) is too similar to Oracle.consult(address,uint256,addr
ess).price1Cumulative (Oracle.sol#466)
Variable Oracle.consult(address,uint256,address).price0Cumulative (Oracle.sol#466) is too similar to Oracle.update(address,addr
ess).price1Cumulative (Oracle.sol#443)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
Oracle.sol analyzed (7 contracts with 84 detectors), 38 result(s) found
```

Slither log >> SwapMining.sol

```
SwapMining.setRouter(address) (SwapMining.sol#2973-2976) should emit an event for:
        - router = newRouter (SwapMining.sol#2975)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

SwapMining.addPair(uint256,address,bool) (SwapMining.sol#2912-2928) should emit an event for:
        - totalAllocPoint = totalAllocPoint.add(_allocPoint) (SwapMining.sol#2918)
SwapMining.setPair(uint256,uint256,bool) (SwapMining.sol#2931-2937) should emit an event for:
        - totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint) (SwapMining.sol#2935)
SwapMining.setArcherswapPerSecond(uint256) (SwapMining.sol#2940-2943) should emit an event for:
        - bowPerSecond = _newPerSecond (SwapMining.sol#2942)
SwapMining.setHalvingPeriod(uint256) (SwapMining.sol#2969-2971) should emit an event for:
        - halvingPeriod = _period (SwapMining.sol#2970)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

SwapMining.constructor(BowToken,IArcherswapFactory,IOracle,address,address,uint256,uint256)._targetToken (SwapMining.sol#2872)
lacks a zero-check on :
                - targetToken = _targetToken (SwapMining.sol#2884)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
Redundant expression "this (SwapMining.sol#1807)" inContext (SwapMining.sol#1801-1810)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

SwapMining.bowToken (SwapMining.sol#2859) should be immutable
SwapMining.factory (SwapMining.sol#2857) should be immutable
SwapMining.startTime (SwapMining.sol#2848) should be immutable
SwapMining.targetToken (SwapMining.sol#2861) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
SwapMining.sol analyzed (14 contracts with 84 detectors), 481 result(s) found
```

Slither log >> SyrupBar.sol

```
Low level call in Address.sendValue(address,uint256) (SyrupBar.sol#52-58):
        - (success) = recipient.call{value: amount}() (SyrupBar.sol#56)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (SyrupBar.sol#131-157):
        - (success,returndata) = target.call{value: weiValue}(data) (SyrupBar.sol#140)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter BowToken.mintFor(address,uint256)._to (SyrupBar.sol#651) is not in mixedCase
Parameter BowToken.mintFor(address,uint256)._amount (SyrupBar.sol#651) is not in mixedCase
Variable BowToken._delegates (SyrupBar.sol#667) is not in mixedCase
Parameter SyrupBar.mint(address,uint256)._to (SyrupBar.sol#892) is not in mixedCase
Parameter SyrupBar.mint(address,uint256)._amount (SyrupBar.sol#892) is not in mixedCase
Parameter SyrupBar.burn(address,uint256)._from (SyrupBar.sol#897) is not in mixedCase
Parameter SyrupBar.burn(address,uint256)._amount (SyrupBar.sol#897) is not in mixedCase
Parameter SyrupBar.safeCakeTransfer(address,uint256)._to (SyrupBar.sol#910) is not in mixedCase
Parameter SyrupBar.safeCakeTransfer(address,uint256)._amount (SyrupBar.sol#910) is not in mixedCase
Variable SyrupBar._delegates (SyrupBar.sol#919) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (SyrupBar.sol#276)" inContext (SyrupBar.sol#270-279)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

SyrupBar.cake (SyrupBar.sol#903) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
SyrupBar.sol analyzed (8 contracts with 84 detectors), 42 result(s) found
```

Slither log >> ArcherswapFactory.sol

```
Redundant expression "this (Factory.sol#247)" inContext (Factory.sol#241-250)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable ArcherswapPair.swap(uint256,uint256,address,bytes).balance0Adjusted (Factory.sol#599) is too similar to ArcherswapPair
.swap(uint256,uint256,address,bytes).balance1Adjusted (Factory.sol#600)
Variable ArcherswapPair.price0CumulativeLast (Factory.sol#445) is too similar to ArcherswapPair.price1CumulativeLast (Factory.s
ol#446)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

ArcherswapFactory.createPair(address,address) (Factory.sol#774-789) uses literals with too many digits:
        - bytecode = type()(ArcherswapPair).creationCode (Factory.sol#779)
ArcherswapFactory.slitherConstructorConstantVariables() (Factory.sol#751-801) uses literals with too many digits:
        - INIT_CODE_PAIR_HASH = keccak256(bytes)(abi.encodePacked(type()(ArcherswapPair).creationCode)) (Factory.sol#752)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

ArcherswapERC20.DOMAIN_SEPARATOR (Factory.sol#313) should be immutable
ArcherswapPair.factory (Factory.sol#437) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
Factory.sol analyzed (14 contracts with 84 detectors), 53 result(s) found
```

Slither log >> ArcherswapPair.sol

```
Low level call in ArcherswapPair._safeTransfer(address,address,uint256) (Pair.sol#463-466):
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Pair.sol#464)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IArcherswapERC20.DOMAIN_SEPARATOR() (Pair.sol#234) is not in mixedCase
Function IArcherswapERC20.PERMIT_TYPEHASH() (Pair.sol#235) is not in mixedCase
Variable ArcherswapERC20.DOMAIN_SEPARATOR (Pair.sol#313) is not in mixedCase
Parameter ArcherswapPair.initialize(address,address)._token0 (Pair.sol#485) is not in mixedCase
Parameter ArcherswapPair.initialize(address,address)._token1 (Pair.sol#485) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (Pair.sol#247)" inContext (Pair.sol#241-250)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable ArcherswapPair.swap(uint256,uint256,address,bytes).balance0Adjusted (Pair.sol#599) is too similar to ArcherswapPair.sw
ap(uint256,uint256,address,bytes).balance1Adjusted (Pair.sol#600)
Variable ArcherswapPair.price0CumulativeLast (Pair.sol#445) is too similar to ArcherswapPair.price1CumulativeLast (Pair.sol#446
)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

ArcherswapERC20.DOMAIN_SEPARATOR (Pair.sol#313) should be immutable
ArcherswapPair.factory (Pair.sol#437) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
Pair.sol analyzed (11 contracts with 84 detectors), 32 result(s) found
```

Slither log >> ArcherswapRouter.sol

```
Function IArcherswapPair.DOMAIN_SEPARATOR() (Router.sol#100) is not in mixedCase
Function IArcherswapPair.PERMIT_TYPEHASH() (Router.sol#101) is not in mixedCase
Function IArcherswapPair.MINIMUM_LIQUIDITY() (Router.sol#118) is not in mixedCase
Function IArcherswapRouter01.WETH() (Router.sol#416) is not in mixedCase
Parameter ArcherswapRouter.setSwapMining(address)._swapMininng (Router.sol#587) is not in mixedCase
Variable ArcherswapRouter.WETH (Router.sol#579) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (Router.sol#11)" inContext (Router.sol#5-14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
```

```
Variable IArcherswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Rout
er.sol#421) is too similar to IArcherswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
.amountBDesired (Router.sol#422)
Variable IArcherswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Rout
er.sol#421) is too similar to ArcherswapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).am
ountBDesired (Router.sol#633)
Variable ArcherswapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Router.
sol#632) is too similar to ArcherswapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amoun
tBDesired (Router.sol#633)
Variable ArcherswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (Router.sol#604) is too
 similar to ArcherswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (Router.sol#605)
Variable IArcherswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Rout
er.sol#421) is too similar to ArcherswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (R
outer.sol#605)
Variable ArcherswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (Router.sol#604) is too
 similar to ArcherswapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (Rout
er.sol#633)
Variable ArcherswapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Router.
sol#632) is too similar to ArcherswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (Rout
er.sol#605)
Variable ArcherswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (Router.sol#604) is too
 similar to IArcherswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (R
outer.sol#422)
Variable ArcherswapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Router.
sol#632) is too similar to IArcherswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).am
ountBDesired (Router.sol#422)
Variable ArcherswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountAOptimal (Router.sol#622) is too
 similar to ArcherswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBOptimal (Router.sol#617)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
Router.sol analyzed (13 contracts with 84 detectors), 39 result(s) found
```

Slither log >> BowToken.sol

```
Address.isContract(address) (BowToken.sol#23-34) uses assembly
        - INLINE ASM (BowToken.sol#30-32)
Address._functionCallWithValue(address,bytes,uint256,string) (BowToken.sol#131-157) uses assembly
        - INLINE ASM (BowToken.sol#149-152)
BowToken.getChainId() (BowToken.sol#879-883) uses assembly
        - INLINE ASM (BowToken.sol#881)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Low level call in Address.sendValue(address,uint256) (BowToken.sol#52-58):
        - (success) = recipient.call{value: amount}() (BowToken.sol#56)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (BowToken.sol#131-157):
        - (success,returndata) = target.call{value: weiValue}(data) (BowToken.sol#140)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter BowToken.mintFor(address,uint256)._to (BowToken.sol#647) is not in mixedCase
Parameter BowToken.mintFor(address,uint256)._amount (BowToken.sol#647) is not in mixedCase
Variable BowToken._delegates (BowToken.sol#663) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (BowToken.sol#276)" inContext (BowToken.sol#270-279)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
BowToken.sol analyzed (7 contracts with 84 detectors), 30 result(s) found
```

Slither log >> BowStakingToken.sol

```
Low level call in Address.sendValue(address,uint256) (xBow.sol#75-81):
        - (success) = recipient.call{value: amount}() (xBow.sol#79)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (xBow.sol#141-162):
        - (success,returndata) = target.call{value: weiValue}(data) (xBow.sol#145)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter BowStakingToken.enter(uint256)._amount (xBow.sol#734) is not in mixedCase
Parameter BowStakingToken.leave(uint256)._share (xBow.sol#755) is not in mixedCase
Function BowStakingToken.BOWBalance(address) (xBow.sol#783-787) is not in mixedCase
Parameter BowStakingToken.BOWBalance(address)._account (xBow.sol#783) is not in mixedCase
Parameter BowStakingToken.xBOWForBOW(uint256)._xBOWAmount (xBow.sol#790) is not in mixedCase
Function BowStakingToken.BOWForxBOW(uint256) (xBow.sol#796-805) is not in mixedCase
Parameter BowStakingToken.BOWForxBOW(uint256)._bowAmount (xBow.sol#796) is not in mixedCase
Parameter BowStakingToken.burn(address,uint256)._from (xBow.sol#843) is not in mixedCase
Parameter BowStakingToken.burn(address,uint256)._amount (xBow.sol#843) is not in mixedCase
Parameter BowStakingToken.mint(address,uint256)._amount (xBow.sol#848) is not in mixedCase
Parameter BowStakingToken.setAdmin(address)._admin (xBow.sol#1072) is not in mixedCase
Variable BowStakingToken._stakedTime (xBow.sol#705) is not in mixedCase
Variable BowStakingToken._delegates (xBow.sol#814) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (xBow.sol#21)" inContext (xBow.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable BowStakingToken.xBOWForBOW(uint256)._xBOWAmount (xBow.sol#790) is too similar to BowStakingToken.BOWForxBOW(uint256).x
BOWAmount_ (xBow.sol#796)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

BowStakingToken.bow (xBow.sol#699) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
xBow.sol analyzed (6 contracts with 84 detectors), 50 result(s) found
```

Slither log >> LakeOfBow.sol

```
Low level call in Address.sendValue(address,uint256) (LakeOfBow.sol#130-136):
        - (success) = recipient.call{value: amount}() (LakeOfBow.sol#134)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (LakeOfBow.sol#196-217):
        - (success,returndata) = target.call{value: weiValue}(data) (LakeOfBow.sol#200)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IUniswapV2ERC20.DOMAIN_SEPARATOR() (LakeOfBow.sol#545) is not in mixedCase
Function IUniswapV2ERC20.PERMIT_TYPEHASH() (LakeOfBow.sol#546) is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (LakeOfBow.sol#567) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (LakeOfBow.sol#568) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (LakeOfBow.sol#585) is not in mixedCase
Parameter LakeOfBow.addAuth(address)._auth (LakeOfBow.sol#675) is not in mixedCase
Parameter LakeOfBow.revokeAuth(address)._auth (LakeOfBow.sol#680) is not in mixedCase
Parameter LakeOfBow.setDevCut(uint256)._amount (LakeOfBow.sol#701) is not in mixedCase
Parameter LakeOfBow.setDevAddr(address)._addr (LakeOfBow.sol#708) is not in mixedCase
Variable LakeOfBow._bridges (LakeOfBow.sol#646) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (LakeOfBow.sol#460)" inContext (LakeOfBow.sol#454-463)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
LakeOfBow.sol analyzed (10 contracts with 84 detectors), 46 result(s) found
```

Slither log >> Multicall.sol

```
Multicall.aggregate(Multicall.Call[]) (Multicall.sol#13-21) has external calls inside a loop: (success,ret) = calls[i].target.c
all(calls[i].callData) (Multicall.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Low level call in Multicall.aggregate(Multicall.Call[]) (Multicall.sol#13-21):
        - (success,ret) = calls[i].target.call(calls[i].callData) (Multicall.sol#17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
Multicall.sol analyzed (1 contracts with 84 detectors), 2 result(s) found
```

Slither log >> WETH9.sol

```
WETH9.decimals (WETH9.sol#7) should be constant
WETH9.name (WETH9.sol#5) should be constant
WETH9.symbol (WETH9.sol#6) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
WETH9.sol analyzed (1 contracts with 84 detectors), 3 result(s) found
```

Solidity Static Analysis

MasterChef.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in MasterChef.safeCakeTransfer(address,uint256,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1684:4:

Gas & Economy

Gas costs:

Gas requirement of function MasterChef.withdraw 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: 1651:4:

Miscellaneous

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: 1722:8:

NFTController.sol

Email: audit@EtherAuthority.io

Gas & Economy

Gas costs:

Gas requirement of function NFTController.transferOwnership 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: 66:4:

Miscellaneous

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: 67:8:

Oracle.sol

Security

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: 465:27:

Gas & Economy

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: 308:8:

Miscellaneous

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: 442:8:

SwapMining.sol

Security

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: 1671:50:

Gas & Economy

Gas costs:

Gas requirement of function SwapMining.getTotalUserReward 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: 3157:4:

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: 2816:8:

Miscellaneous

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: 2486:12:

SyrupBar.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SyrupBar.safeCakeTransfer(address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1039:4:

Gas & Economy

Gas costs:

Gas requirement of function SyrupBar.getPriorVotes 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: 1180:4:

Miscellaneous

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: 1294:8:

ArcherswapFactory.sol

Security

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: 514:44:

Gas & Economy

Gas costs:

Gas requirement of function YumiswapFactory.createPair 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: 774:4:

Miscellaneous

Similar variable names:

YumiswapFactory.createPair(address,address) : Variables have very similar names "token0" and "tokenA". Note: Modifiers are currently not considered by this static analysis.

Pos: 785:16:

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: 778:8:

ArcherswapPair.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in YumiswapPair._mintFee(uint112,uint112): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 558:4:

Gas & Economy

Gas costs:

Gas requirement of function YumiswapERC20.name 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: 356:4:

Miscellaneous

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: 638:8:

ArcherswapRouter.sol

Security

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: 409:26:

Email: audit@EtherAuthority.io

Gas & Economy

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: 909:25:

Miscellaneous

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: 925:8:

BowToken.sol

Security

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: 759:16:

Gas & Economy

Gas costs:

Gas requirement of function BowToken.getPriorVotes 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: 784:4:

Miscellaneous

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: 759:8:

BowStakingToken.sol

Security

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: 1067:8:

Gas & Economy

Gas costs:

Gas requirement of function BowStakingToken.getPriorVotes 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: 970:4:

Miscellaneous

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: 729:8:

LakeOfBow.sol

Security

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases. If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.

more

Pos: 726:30:

Gas & Economy

Gas costs:

Gas requirement of function LakeOfBow.bridgeFor 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: 716:4:

Miscellaneous

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: 912:8:

Multicall.sol

Security

Block hash:

Use of "blockhash": "blockhash(uint blockNumber)" is used to access the last 256 block hashes. A miner computes the block hash by "summing up" the information in the current block mined. By "summing up" the information cleverly, a miner can try to influence the outcome of a transaction in the current block. This is especially easy if there are only a small number of equally likely outcomes.

Pos: 30:20:

Email: audit@EtherAuthority.io

Gas & Economy

Gas costs:

Gas requirement of function Multicall.aggregate 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: 13:4:

Miscellaneous

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: 18:12:

WETH9.sol

Gas & Economy

Gas costs:

Gas requirement of function WETH9.withdraw 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: 21:4:

Solhint Linter

MasterChef.sol

```
MasterChef.sol:3:1: Error: Compiler version >=0.6.12 does not satisfy
the r semver requirementMasterChef.sol:1367:20: Error: Variable name
must be in mixedCase
MasterChef.sol:1379:29: Error: Constant name must be in capitalized
SNAKE_CASE
MasterChef.sol:1620:31: Error: Avoid to make time-based decisions in
your business logic
```

NFTController.sol

```
NFTController.sol:3:1: Error: Compiler version 0.6.12 does not
satisfy the r semver requirement
NFTController.sol:79:26: Error: Code contains empty blocks
```

Oracle.sol

```
Oracle.sol:3:1: Error: Compiler version >=0.6.6 does not satisfy the
r semver requirementOracle.sol:56:5: Error: Function name must be in
mixedCaseliteralsOracle.sol:335:5: Error: Contract name must be in
CamelCase
Oracle.sol:353:25: Error: Use double quotes for string literals
Oracle.sol:441:28: Error: Avoid to make time-based decisions in your
business logic
Oracle.sol:442:39: Error: Use double quotes for string
literalsOracle.sol:465:28: Error: Avoid to make time-based decisions
in your business logic
```

SwapMining.sol

```
SwapMining.sol:3:1: Error: Compiler version >=0.6.0 <0.8.0 does not
satisfy the r semver requirement
SwapMining.sol:5:1: Error: Contract name must be in CamelCase
SwapMining.sol:6:2: Error: Explicitly mark visibility of state
SwapMining.sol:11:3: Error: Avoid using inline assembly. It is
acceptable only in rare cases
SwapMining.sol:13:8: Error: Variable "r" is unused
SwapMining.sol:1584:50: Error: Use double quotes for string literals
SwapMining.sol:2175:48: Error: Use double quotes for string literals
SwapMining.sol:2290:17: Error: Avoid to make time-based decisions in
your business logic
```

```
SwapMining.sol:2412:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
SwapMining.sol:2737:5: Error: Function name must be in mixedCase
SwapMining.sol:3130:39: Error: Avoid to make time-based decisions in
your business logic
```

SyrupBar.sol

```
SyrupBar.sol:3:1: Error: Compiler version >=0.6.12 does not satisfy
the r semver requirement
SyrupBar.sol:648:48: Error: Use double quotes for string literals
SyrupBar.sol:1022:17: Error: Avoid to make time-based decisions in
your business logic
SyrupBar.sol:1164:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
```

ArcherswapFactory.sol

```
ArcherswapFactory.sol:3:1: Error: Compiler version >=0.6.6 does not
satisfy the r semver requirement
ArcherswapFactory.sol:655:5: Error: Function name must be in
mixedCase
ArcherswapFactory.sol:778:56: Error: Use double quotes for string
literals
ArcherswapFactory.sol:781:9: Error: Avoid using inline assembly. It
is acceptable only in rare cases
ArcherswapFactory.sol:797:44: Error: Use double quotes for string
literals
```

ArcherswapPair.sol

```
ArcherswapPair.sol:3:1: Error: Compiler version >=0.6.6 does not
satisfy the r semver requirement
ArcherswapPair.sol:379:29: Error: Avoid to make time-based decisions
in your business logic
ArcherswapPair.sol:494:40: Error: Avoid to make time-based decisions
in your business logic
ArcherswapPair.sol:601:104: Error: Use double quotes for string
literals
```

ArcherswapRouter.sol

```
ArcherswapRouter.sol:3:1: Error: Compiler version >=0.6.6 does not
satisfy the r semver requirement
ArcherswapRouter.sol:583:29: Error: Avoid to make time-based
```

```
decisions in your business logic
ArcherswapRouter.sol:583:46: Error: Use double quotes for string
literals
ArcherswapRouter.sol:591:35: Error: Variable name must be in
mixedCase
ArcherswapRouter.sol:971:44: Error: Use double quotes for string
literals
```

BowToken.sol

```
BowToken.sol:4:1: Error: Compiler version >=0.4.0 does not satisfy
the r semver requirement
BowToken.sol:644:48: Error: Use double quotes for string literals
BowToken.sol:759:17: Error: Avoid to make time-based decisions in
your business logic
BowToken.sol:881:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
```

BowStakingToken.sol

```
BowStakingToken.sol:3:1: Error: Compiler version 0.6.12 does not
satisfy the r semver requirement
BowStakingToken.sol:536:94: Error: Code contains empty blocks
BowStakingToken.sol:722:57: Error: Avoid to make time-based decisions
in your business logic
BowStakingToken.sol:796:5: Error: Function name must be in mixedCase
BowStakingToken.sol:945:17: Error: Avoid to make time-based decisions
in your business logic
BowStakingToken.sol:1067:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
```

LakeOfBow.sol

```
LakeOfBow.sol:4:1: Error: Compiler version 0.6.12 does not satisfy
the r semver requirement
LakeOfBow.sol:585:5: Error: Function name must be in mixedCase
LakeOfBow.sol:726:31: Error: Avoid to use tx.origin
LakeOfBow.sol:911:31: Error: Use double quotes for string literals
LakeOfBow.sol:912:50: Error: Use double quotes for string literals
```

Multicall.sol

```
Multicall.sol:3:1: Error: Compiler version >=0.5.0 does not satisfy
the r semver requirement
Multicall.sol:17:48: Error: Avoid using low level calls.
```

```
Multicall.sol:33:21: Error: Avoid to make time-based decisions in
your business logic
```

WETH9.sol

```
WETH9.sol:3:1: Error: Compiler version ^0.6.12 does not satisfy the r
semver requirement
WETH9.sol:27:55: Error: Code contains empty blocks
```

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