SMART CONTRACT CODE SECURITY ANALYSIS REPORT

Customer: Hakuswap Protocol

Platform: Avalanche

Language: Solidity

Date: December 24th, 2021

Table of Content

Disclaimer..... 3
Purpose of the report 3
Introduction.....4
Audit Summary 5
Overview.....6
Methodology.....6
Classification / Issue Types Definition:.....6
Attacks & Issues considered while auditing 7
Overflows and underflows.....7
Reentrancy Attack.....7
Replay attack.....7
Short address attack 8
Approval Double-spend 8
Sybil attacks 8
Issues Found 9
High Severity Issues.....9
Moderate Severity Issues.....9
Medium Severity Issues..... 9
Low Severity Issues..... 9
Informational Observations.....10
Audit Conclusion 14
Appendix 15
Smart Contract Functional Summary.....15
Code Flow Diagram 24
Slither Results Log 35

Copyright © 2021
Hash0X
(audit@hash0x.com)

Disclaimer

Hash0X reports are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Hash0X to perform a security review.

The audit makes no statements or warranties about the utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The content of this audit report is provided “as is”, without representations and warranties of any kind, and Hash0X disclaims any liability for damage arising out of, or in connection with, this audit report. Copyright of this report remains with Hash0X.

Purpose of the report

The Audits and the analysis described therein are created solely for Clients and published with their consent. The scope of our review is limited to a review of Solidity code and only the Solidity code we note as being within the scope of our review within this report. The Solidity language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the Solidity programming language that could present security risks. Cryptographic tokens and smart contracts are emergent technologies and carry with them high levels of technical risk and uncertainty.

The Audits are not an endorsement or indictment of any particular project or team, and the Audits do not guarantee the security of any particular project. This Report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset.

Copyright © 2021
Hash0X
(audit@hash0x.com)

Introduction

We first thank the Hakuswap Team for giving us the opportunity to audit their smart contract. This document outlines our methodology, audit details, and results. The Hakuswap Protocol Team asked us to review their Hakuswap Protocol smart contract.

Hash0X reviewed the system from a technical perspective looking for bugs, issues and vulnerabilities in their code base. This audit report is valid for the smart contracts at the mentioned commit hashes only. This audit report is not valid for any other versions of the smart contracts files.

Project file

File Name	Updated File MD5 Hash
HakuToken.sol	7BBB5A8ACEB800412B4EC46211F40B81
Syrupbar.sol	3E4A741FBAE6CBDDB39F1917E09B3152
MasterChef.sol	99DAD814F4A862F12BE921D74C41D019
HakuswapFactory.sol	99DAD814F4A862F12BE921D74C41D019
HakuswapRouter.sol	A083EAB9F47C572F0860459F9192CCBD
Oracle.sol	E008B24B88E1B8788F39BDB90C9A7961
SwapMining.sol	8D8A428D7D67D2C0C9C809870B4DEEE3
xHAKU.sol	F238C977B9A1206553D615FF7C00E8FE
LakeOfHaku.sol	0513B5356AC414458DA9F8B284EAC329
Multicall.sol	CD78A297F742B45105931F70C0458053

Copyright © 2021
Hash0X
(audit@hash0x.com)

Audit Summary

	Total	Resolved	UnResolved	Acknowledged
High Severity Issues Found	0	0	0	0
Moderate Severity Issues Found	0	0	0	0
Medium Severity Issues	0	0	0	0
Low Severity Issues	1	1	0	0
Informational Observations	2	1	0	1

The smart contract is considered to **pass the audit**, as of the audit date, if no high severity or moderate severity issues are found.

Copyright © 2021
Hash0X
(audit@hash0x.com)

Overview

The project has 13 smart contract codes in the form of a github code. Solidity files for the Hakuswap Protocol Smart Contract. This Protocol has smart contracts which interact with each other in a complex way. We manually reviewed each line of code in the smart contracts within the scope.

Methodology

Hash0X manually reviewed the smart contracts line-by-line, keeping in mind industry best practices and known attacks, looking for any potential issues and vulnerabilities, and areas where improvements are possible.

We also used automated tools like slither / surya for analysis and reviewing the smart contracts. These tools often give false-positives, and any issues reported by them but not included in the issue list can be considered not valid.

Classification / Issue Types Definition:

1. **High Severity**: which presents a significant security vulnerability or failure of the contract across a range of scenarios, or which may result in loss of funds.
2. **Moderate Severity**: which affects the desired outcome of the contract execution or introduces a weakness that can be exploited. It may not result in loss of funds but breaks the functionality or produces unexpected behavior.
3. **Low Severity**: which does not have a material impact on the contract execution and is likely to be subjective.

As mentioned above, the smart contract is considered to pass the audit, as of the audit date, if no high severity or moderate severity issues are found.

Copyright © 2021
Hash0X
(audit@hash0x.com)

Attacks & Issues considered while auditing

In order to check for the security of the contract, we reviewed each line of code in the smart contract considering several known Smart Contract Attacks & known issues

- **Overflows and underflows**

  An overflow happens when the limit of the type variable uint256 , 2 ** 256, is exceeded. What happens is that the value resets to zero instead of incrementing more. For instance, if we want to assign a value to a uint bigger than 2 ** 256 it will simple go to 0—this is dangerous. On the other hand, an underflow happens when you try to subtract 0 minus a number bigger than 0. For example, if you subtract 0 - 1 the result will be = 2 ** 256 instead of -1. This is quite dangerous. This contract DOES check for overflows and underflows using SafeMath libraries.

- **Reentrancy Attack**

  One of the major dangers of calling external contracts is that they can take over the control flow, and make changes to your data that the calling function wasn't expecting. This class of bug can take many forms, and both of the major bugs that led to the DAO's collapse were bugs of this sort. This smart contract uses a Check-effect pattern to protect against this attack.

- **Replay attack**

  The replay attack consists of making a transaction on one blockchain like the original Ethereum’s blockchain and then repeating it on another blockchain like Ethereum's classic blockchain. The ether is transferred like a normal transaction from a blockchain to another. Though it's no longer a problem because the version 1.5.3 of Geth and 1.4.4 of Parity both implement the attack protection EIP 155 by Vitalik Buterin. So, the people that will use the contract depend on their own ability to be updated with those programs to keep themselves secure. Since this full system is a cross-chain bridge between Binance Smart Chain and Ethereum Blockchains – it is recommended to not let users enter arbitrary chain IDs in the transfer and receipt requests which may result in a potential replay attack in the future.

Copyright © 2021
Hash0X
(audit@hash0x.com)

- **Short address attack**

  This attack affects ERC20 tokens, was discovered by the Golem team and consists of the following: A user creates an Ethereum wallet with a trailing 0, which is not hard because it's only a digit. For instance: 0xiofa8d97756as7df5sd8f75g8675ds8gsdg0 Then he buys tokens by removing the last zero: Buy 1000 tokens from account 0xiofa8d97756as7df5sd8f75g8675ds8gsdg. If the contract has enough tokens and the but function doesn't check the length of the address of the sender, Ethereum's virtual machine will just add zeros to the transaction until the address is complete. This issue is not applicable to JetSwap smart contracts.

- **Approval Double-spend**

  The ERC20 Standard allows users to approve other users to manage their tokens, or spend tokens from their account till a certain amount, by setting the user's allowance with the standard `approve` function, then the allowed user may use `transferFrom` to spend the allowed tokens. Hypothetically, given a situation where Alice approves Bob to spend 100 Tokens from her account, and if Alice needs to adjust the allowance to allow Bob to spend 20 more tokens, normally – she'd check Bob's allowance (100 currently) and start a new `approve` transaction allowing Bob to spend a total of 120 Tokens instead of 100 Tokens.

  Likely impact of this bug is low for most situations. For more, see this discussion on GitHub: https://github.com/ethereum/EIPs/issues/20#issuecomment263524729

- **Sybil attacks**

  In a Sybil attack, the attacker subverts the reputation system of a network service by creating a large number of pseudonymous identities and uses them to gain a disproportionately large influence. Normally in BEP20 & DeFi smart contracts, sybil attacks are related to voting power amplification where a user may use their vote more than once in case of governance tokens. Proper user input restrictions in this contract prevented this type of attack.

Copyright © 2021
Hash0X
(audit@hash0x.com)

Issues Found

High Severity Issues

No High severity vulnerabilities were found.

Moderate Severity Issues

No Moderate severity vulnerabilities were found.

Medium Severity Issues

No Medium severity vulnerabilities were found.

Low Severity Issues

(1) Same LPToken can be added more than once: **- MasterChef.sol**

Owner can add the same lptoken using the add() function. But as per the comment, that should not be done.

Resolution:

We suggest adding validation for duplicate lptoken.

Statue: Fixed.

Informational / Best practice Observations

(1) Use the latest solidity version: **- HakuToken.sol, Syrupbar.sol, MasterChef.sol**

```
pragma solidity >0.6.6;
```

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

Resolution:

We suggest using 0.8.10 which is the latest version.

Status: Acknowledged.

(2) SPDX license identifier: **- Multicall.sol**

SPDX license identifier not provided in source file.

Resolution:

Please add SPDX-License-Identifier.

Status: Fixed.

Copyright © 2021
Hash0X
(audit@hash0x.com)

Audit Conclusion

The Hakuswap Protocol smart contracts codes were written so systematically that we did not find any major issues in it. Hence, **this code is ready for production**.

Due to the nature of the smart contract Token, there are unlimited use case scenarios, thus it is not possible to give guarantees about the future outcomes. This audit is based on manual code analysis as well as using the latest static tools.

This audit report presents all the findings based on standard audit procedure, which includes manual analysis as well as automated software tools. Smart Contract's high-level description of functions was presented in the Appendix 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 extensive audit procedure scope is "**Well Secured**".

Copyright © 2021
Hash0X
(audit@hash0x.com)

Appendix

Smart Contracts Functional Summary

MasterChef.sol

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	Passed	No Issue
2	updateMultiplier	write	access only Owner	No Issue
3	poolLength	external	Passed	No Issue
4	add	write	access only Owner	No Issue
5	set	write	access only Owner	No Issue
6	updateStakingPool	internal	Passed	No Issue
7	getMultiplier	read	Passed	No Issue
8	pendingCake	external	Passed	No Issue
9	massUpdatePools	write	Passed	No Issue
10	updatePool	write	Passed	No Issue
11	deposit	write	Passed	No Issue
12	withdraw	write	Passed	No Issue
13	enterStaking	write	Passed	No Issue
14	leaveStaking	write	Passed	No Issue
15	emergencyWithdraw	write	Passed	No Issue
16	safeCakeTransfer	internal	Passed	No Issue
17	setCakePerSecond	external	access only Owner	No Issue
18	dev	write	Passed	No Issue
19	setReserveaddr	write	Passed	No Issue
20	setHAKUPoolWeight	write	access only Owner	No Issue
21	owner	read	Passed	No Issue
22	onlyOwner	modifier	Passed	No Issue
23	renounceOwnership	write	access only Owner	No Issue
24	transferOwnership	write	access only Owner	No Issue

Syrupbar.sol

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	Passed	No Issue
2	mint	write	Passed	No Issue
3	burn	write	access only Owner	No Issue
4	safeCakeTransfer	write	access only Owner	No Issue
5	delegates	external	Passed	No Issue
6	delegate	external	Passed	No Issue
7	delegateBySig	external	Passed	No Issue
8	getCurrentVotes	external	Passed	No Issue

Copyright © 2021
Hash0X
(audit@hash0x.com)

9	getPriorVotes	external	Passed	No Issue
10	_delegate	internal	Passed	No Issue
11	_moveDelegates	internal	Passed	No Issue
12	_writeCheckpoint	internal	Passed	No Issue
13	safe32	internal	Passed	No Issue
14	getChainId	internal	Passed	No Issue
15	getOwner	external	Passed	No Issue
16	name	read	Passed	No Issue
17	decimals	read	Passed	No Issue
18	symbol	read	Passed	No Issue
19	totalSupply	read	Passed	No Issue
20	balanceOf	read	Passed	No Issue
21	transfer	write	Passed	No Issue
22	allowance	read	Passed	No Issue
23	approve	write	Passed	No Issue
24	transferFrom	write	Passed	No Issue
25	increaseAllowance	write	Passed	No Issue
26	decreaseAllowance	write	Passed	No Issue
27	_transfer	internal	Passed	No Issue
28	_mint	internal	Passed	No Issue
29	_burn	internal	Passed	No Issue
30	_approve	internal	Passed	No Issue
31	_burnFrom	internal	Passed	No Issue
32	owner	read	Passed	No Issue
33	onlyOwner	modifier	Passed	No Issue
34	renounceOwnership	write	access only Owner	No Issue
35	transferOwnership	write	access only Owner	No Issue

Factory.sol

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

Router.sol

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

Copyright © 2021
Hash0X
(audit@hash0x.com)

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

Copyright © 2021
Hash0X
(audit@hash0x.com)

28	getAmountsIn	read	Passed	No Issue

HakuToken.sol

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	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
14	getOwner	external	Passed	No Issue
15	name	read	Passed	No Issue
16	decimals	read	Passed	No Issue
17	symbol	read	Passed	No Issue
18	totalSupply	read	Passed	No Issue
19	balanceOf	read	Passed	No Issue
20	transfer	write	Passed	No Issue
21	allowance	read	Passed	No Issue
22	approve	write	Passed	No Issue
23	transferFrom	write	Passed	No Issue
24	increaseAllowance	write	Passed	No Issue
25	decreaseAllowance	write	Passed	No Issue
26	mint	write	access only Owner	No Issue
27	_transfer	internal	Passed	No Issue
28	_mint	internal	Passed	No Issue
29	_burn	internal	Passed	No Issue
30	_approve	internal	Passed	No Issue
31	_burnFrom	internal	Passed	No Issue
32	owner	read	Passed	No Issue
33	onlyOwner	modifier	Passed	No Issue
34	renounceOwnership	write	access only Owner	No Issue
35	transferOwnership	write	access only Owner	No Issue

xHAKU.sol

Copyright © 2021
Hash0X
(audit@hash0x.com)

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	Passed	No Issue
2	enter	write	Passed	No Issue
3	leave	write	Passed	No Issue
4	HAKUBalance	external	Passed	No Issue
5	xHAKUForHAKU	external	Passed	No Issue
6	HAKUForxHAKU	external	Passed	No Issue
7	burn	write	Passed	No Issue
8	mint	write	Passed	No Issue
9	transferFrom	write	Passed	No Issue
10	transfer	write	Passed	No Issue
11	_initDelegates	internal	Passed	No Issue
12	delegates	external	Passed	No Issue
13	delegate	external	Passed	No Issue
14	delegateBySig	external	Passed	No Issue
15	getCurrentVotes	external	Passed	No Issue
16	getPriorVotes	external	Passed	No Issue
17	_delegate	internal	Passed	No Issue
18	_moveDelegates	internal	Passed	No Issue
19	_writeCheckpoint	internal	Passed	No Issue
20	safe32	internal	Passed	No Issue
21	getChainId	internal	Passed	No Issue
22	getOwner	external	Passed	No Issue
23	name	read	Passed	No Issue
24	decimals	read	Passed	No Issue
25	symbol	read	Passed	No Issue
26	totalSupply	read	Passed	No Issue
27	balanceOf	read	Passed	No Issue
28	transfer	write	Passed	No Issue
29	allowance	read	Passed	No Issue
30	approve	write	Passed	No Issue
31	transferFrom	write	Passed	No Issue
32	increaseAllowance	write	Passed	No Issue
33	decreaseAllowance	write	Passed	No Issue
34	mint	write	access only Owner	No Issue
35	_transfer	internal	Passed	No Issue
36	_mint	internal	Passed	No Issue
37	_burn	internal	Passed	No Issue
38	_approve	internal	Passed	No Issue
39	_burnFrom	internal	Passed	No Issue
40	owner	read	Passed	No Issue
41	onlyOwner	modifier	Passed	No Issue
42	renounceOwnership	write	access only Owner	No Issue
43	transferOwnership	write	access only Owner	No Issue

Copyright © 2021
Hash0X
(audit@hash0x.com)

LakeOfHaku.sol

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

Multicall.sol

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	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

Oracle.sol

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

Copyright © 2021
Hash0X
(audit@hash0x.com)

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

SwapMining.sol

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	Passed	No Issue
2	poolLength	read	Passed	No Issue
3	addPair	write	access only Owner	No Issue
4	setPair	write	access only Owner	No Issue
5	setHakuswapPerSecond	write	access only Owner	No Issue
6	addWhitelist	write	access only Owner	No Issue
7	delWhitelist	write	access only Owner	No Issue
8	getWhitelistLength	read	Passed	No Issue
9	isWhitelist	read	Passed	No Issue
10	getWhitelist	read	Passed	No Issue
11	setHalvingPeriod	write	access only Owner	No Issue
12	setRouter	write	access only Owner	No Issue
13	setOracle	write	access only Owner	No Issue
14	phase	read	Passed	No Issue
15	phase	read	Passed	No Issue
16	reward	read	Passed	No Issue
17	getHakuReward	read	Passed	No Issue
18	massMintPools	write	Passed	No Issue
19	mint	write	Passed	No Issue
20	onlyRouter	modifier	Passed	No Issue
21	swap	write	access only Router	No Issue
22	getQuantity	write	Passed	No Issue
23	takerWithdraw	write	Passed	No Issue
24	getUserReward	read	Passed	No Issue
25	getPoolInfo	read	Passed	No Issue
26	ownerWithdraw	write	access only Owner	No Issue
27	safeHakuTransfer	internal	Passed	No Issue
28	owner	read	Passed	No Issue
29	onlyOwner	modifier	Passed	No Issue
30	renounceOwnership	write	access only Owner	No Issue
31	transferOwnership	write	access only Owner	No Issue

MockToken.sol

Sl.	Functions	Type	Observation	Conclusion

Copyright © 2021
Hash0X
(audit@hash0x.com)

1	constructor	read	Passed	No Issue
2	mint	write	Passed	No Issue
3	getOwner	external	Passed	No Issue
4	name	read	Passed	No Issue
5	decimals	read	Passed	No Issue
6	symbol	read	Passed	No Issue
7	totalSupply	read	Passed	No Issue
8	balanceOf	read	Passed	No Issue
9	transfer	write	Passed	No Issue
10	allowance	read	Passed	No Issue
11	approve	write	Passed	No Issue
12	transferFrom	write	Passed	No Issue
13	increaseAllowance	write	Passed	No Issue
14	decreaseAllowance	write	Passed	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	owner	read	Passed	No Issue
21	onlyOwner	modifier	Passed	No Issue
22	renounceOwnership	write	access only Owner	No Issue
23	transferOwnership	write	access only Owner	No Issue

Pair.sol

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	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
13	_mint	internal	Passed	No Issue
14	_burn	internal	Passed	No Issue
15	_approve	write	Passed	No Issue
16	_transfer	write	Passed	No Issue
17	approve	external	Passed	No Issue

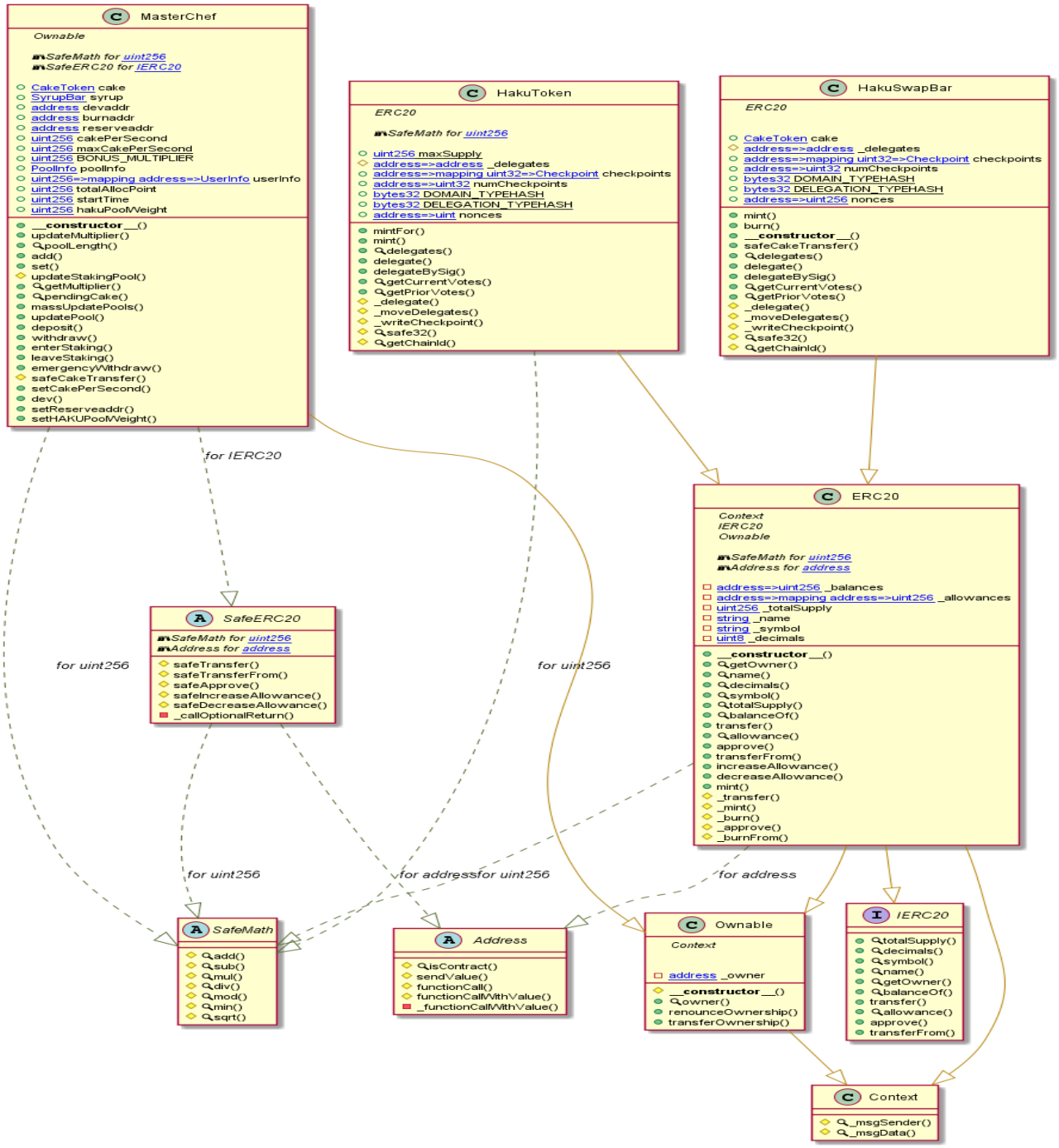
Copyright © 2021
Hash0X
(audit@hash0x.com)

18	transfer	external	Passed	No Issue
19	transferFrom	external	Passed	No Issue
20	permit	external	Passed	No Issue

Copyright © 2021
Hash0X
(audit@hash0x.com)

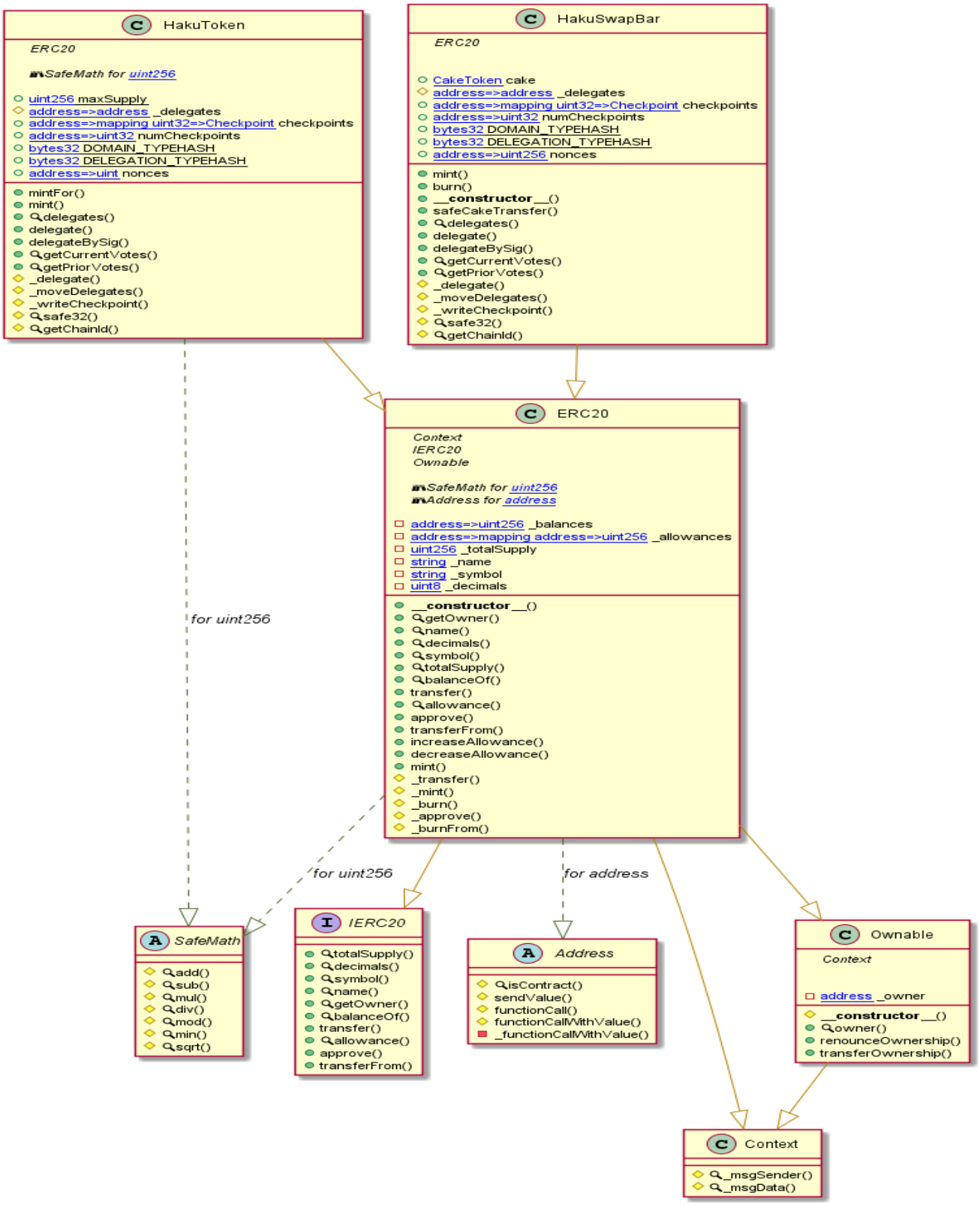
Code Flow Diagram - Hakuswap Protocol

MasterChef Diagram



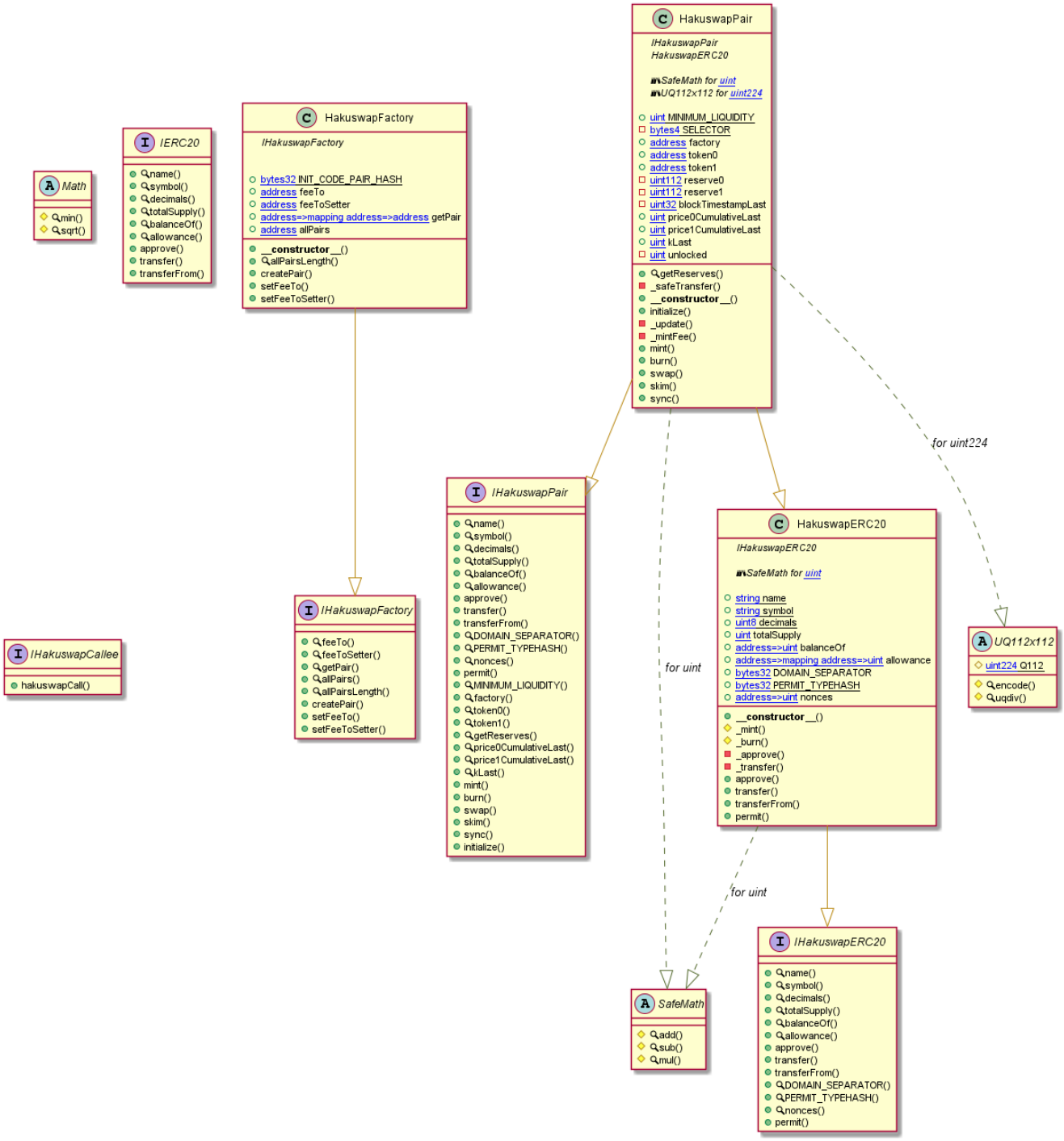
Copyright © 2021
Hash0X
(audit@hash0x.com)

Syrupbar Diagram



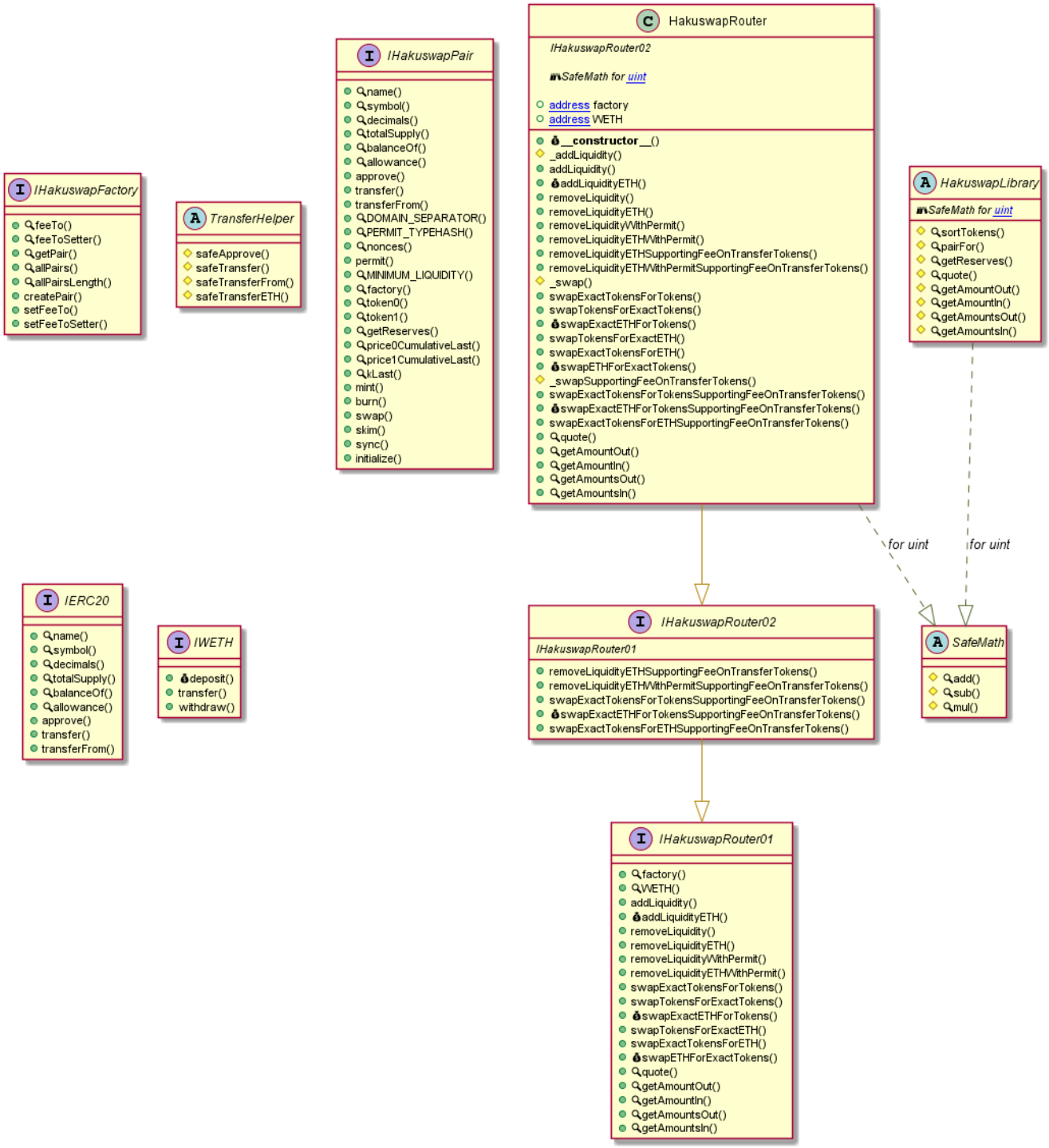
Copyright © 2021
Hash0X
(audit@hash0x.com)

Factory Diagram



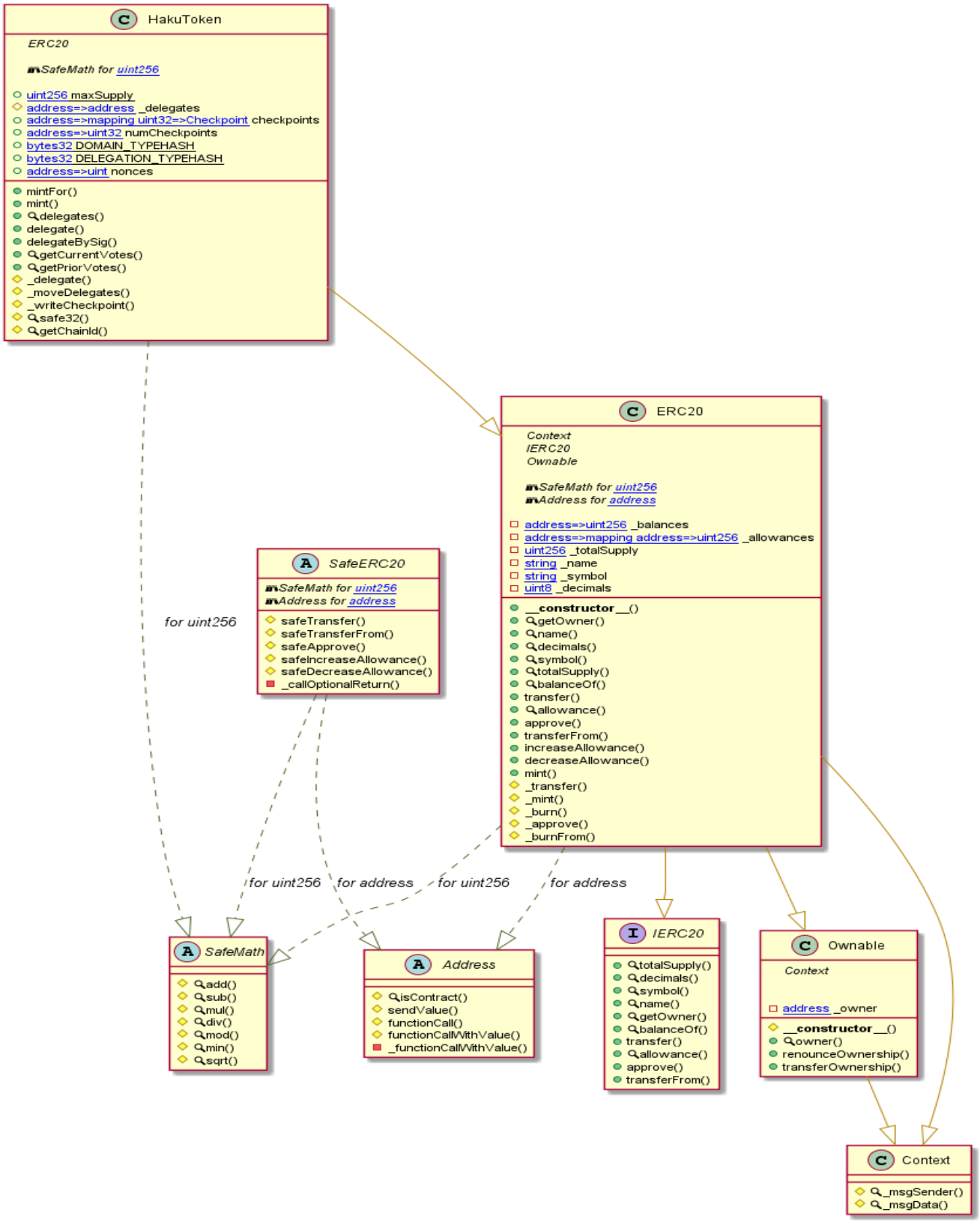
Copyright © 2021
Hash0X
(audit@hash0x.com)

Router Diagram



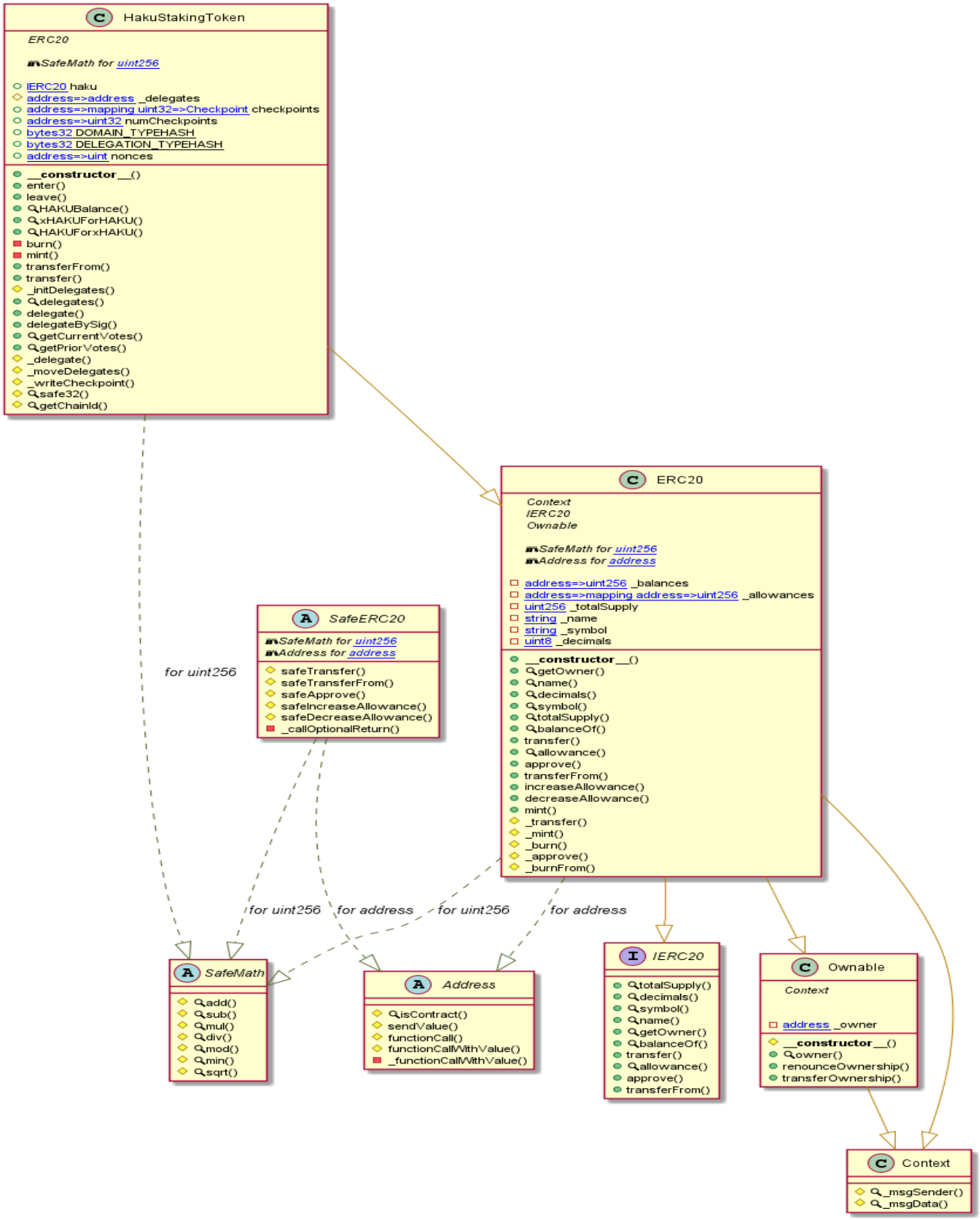
Copyright © 2021
Hash0X
(audit@hash0x.com)

HakuToken Diagram



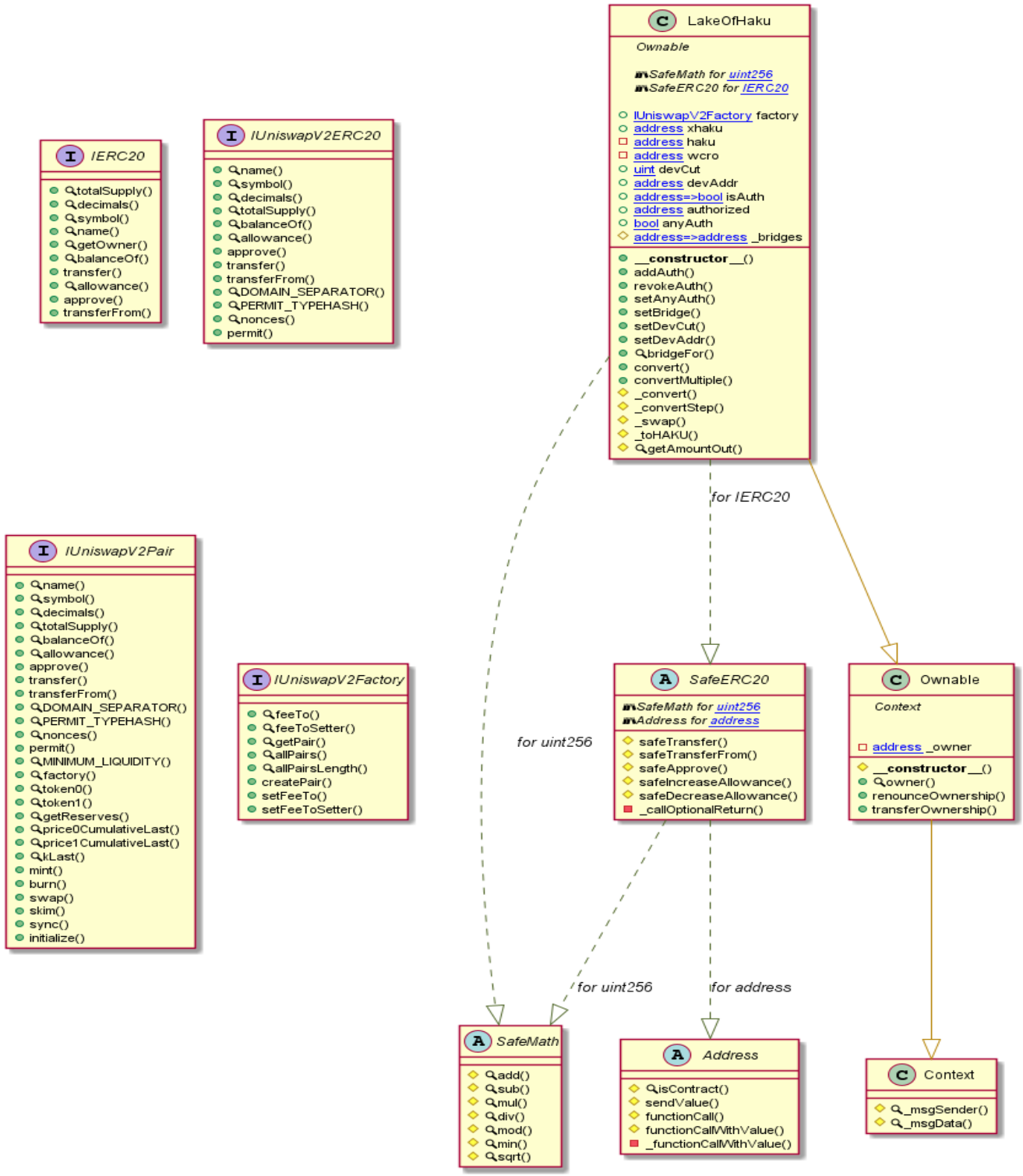
Copyright © 2021
Hash0X
(audit@hash0x.com)

xHAKU Diagram



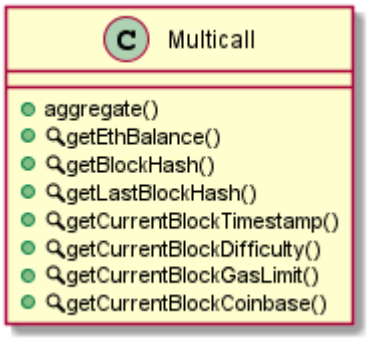
Copyright © 2021
Hash0X
(audit@hash0x.com)

LakeOfHaku Diagram

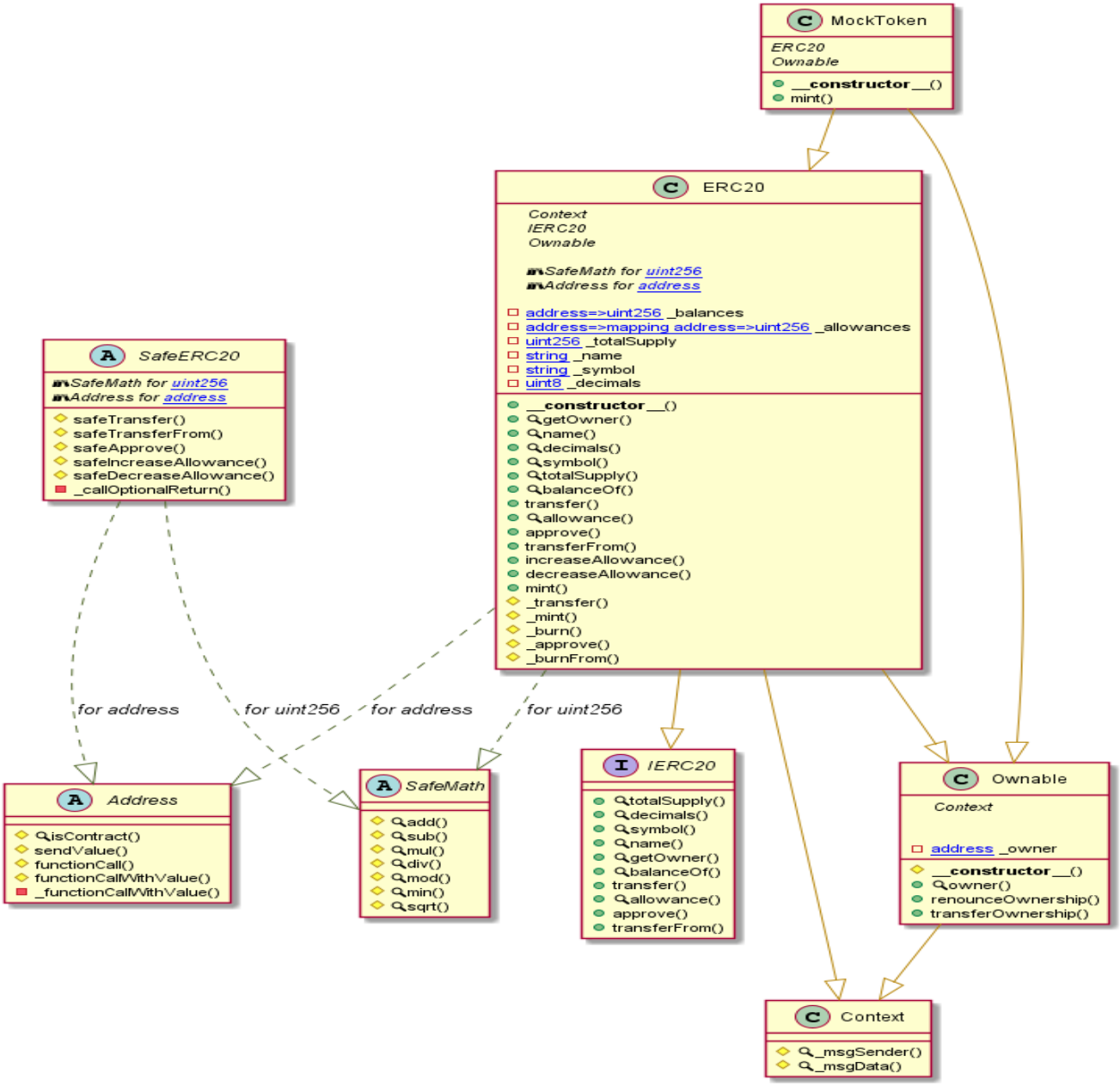


Copyright © 2021
Hash0X
(audit@hash0x.com)

Multicall Diagram

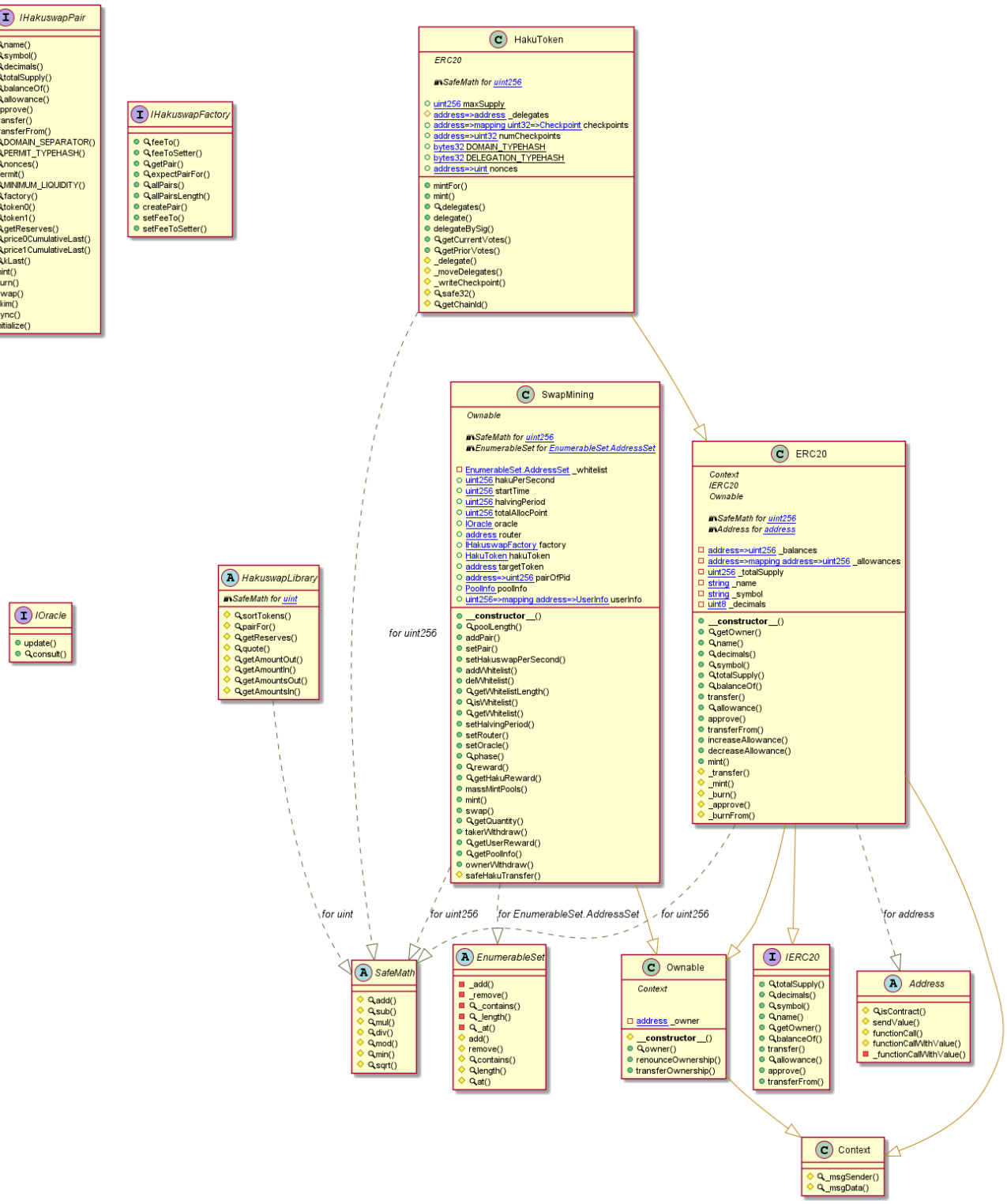


MockToken Diagram



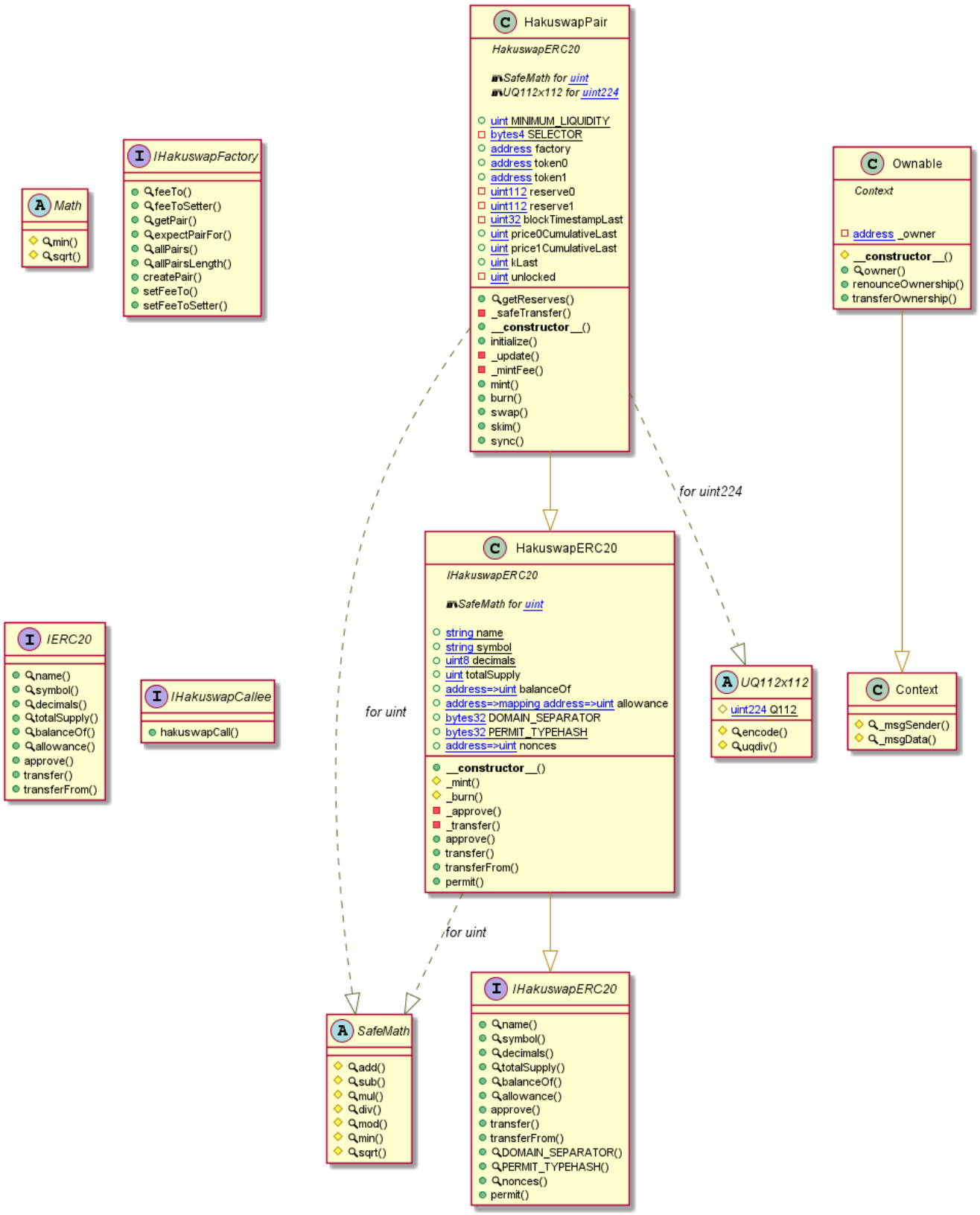
Copyright © 2021
Hash0X
(audit@hash0x.com)

SwapMining Diagram



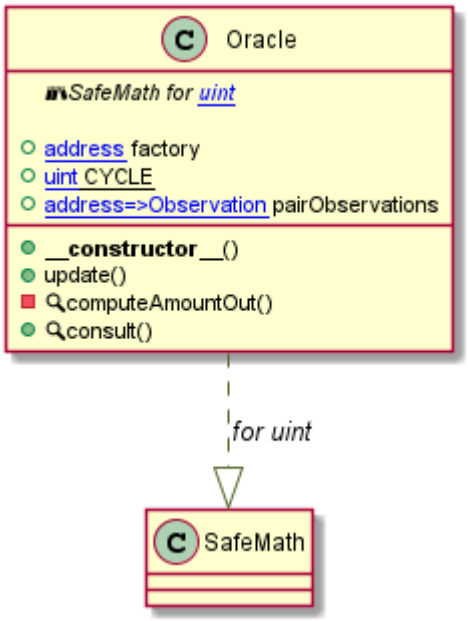
Copyright © 2021
Hash0X
(audit@hash0x.com)

Pair Diagram



Copyright © 2021
Hash0X
(audit@hash0x.com)

Oracle Diagram



Copyright © 2021
Hash0X
(audit@hash0x.com)

Slither Results Log

Slither log >> MasterChef.sol

```
INFO:Detectors:
MasterChef.startTime (MasterChef.sol#917) is never initialized. It is used in:
        - MasterChef.add(uint256,IERC20,bool) (MasterChef.sol#942-962)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
MasterChef.pendingCake(uint256,address) (MasterChef.sol#1008-1029) performs a multiplication on the result of a division:
        -cakeReward = multiplier.mul(cakePerSecond).mul(pool.allocPoint).div(totalAllocPoint) (MasterChef.sol#1020-1023)
        -accCakePerShare = accCakePerShare.add(cakeReward.mul(1e12).div(lpSupply)) (MasterChef.sol#1024-1026)
MasterChef.updatePool(uint256) (MasterChef.sol#1040-1068) performs a multiplication on the result of a division:
        -cakeReward = multiplier.mul(cakePerSecond).mul(pool.allocPoint).div(totalAllocPoint) (MasterChef.sol#1051-1054)
        -pool.accCakePerShare = pool.accCakePerShare.add(cakeReward.mul(1e12).div(lpSupply)) (MasterChef.sol#1064-1066)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
MasterChef.updatePool(uint256) (MasterChef.sol#1040-1068) uses a dangerous strict equality:
        - lpSupply == 0 (MasterChef.sol#1046)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in MasterChef.deposit(uint256,uint256) (MasterChef.sol#1071-1096):
        External calls:
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (MasterChef.sol#1087-1091)
        State variables written after the call(s):
        - user.amount = user.amount.add(_amount) (MasterChef.sol#1092)
        - user.rewardDebt = user.amount.mul(pool.accCakePerShare).div(1e12) (MasterChef.sol#1094)
Reentrancy in MasterChef.emergencyWithdraw(uint256) (MasterChef.sol#1173-1180):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),user.amount) (MasterChef.sol#1176)
        State variables written after the call(s):
        - user.amount = 0 (MasterChef.sol#1178)
        - user.rewardDebt = 0 (MasterChef.sol#1179)
Reentrancy in MasterChef.enterStaking(uint256) (MasterChef.sol#1122-1147):
        External calls:
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (MasterChef.sol#1136-1140)
        State variables written after the call(s):
        - user.amount = user.amount.add(_amount) (MasterChef.sol#1141)
        - user.rewardDebt = user.amount.mul(pool.accCakePerShare).div(1e12) (MasterChef.sol#1143)
Reentrancy in MasterChef.leaveStaking(uint256) (MasterChef.sol#1150-1170):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (MasterChef.sol#1164)
```

```
        State variables written after the call(s):
        - user.rewardDebt = user.amount.mul(pool.accCakePerShare).div(1e12) (MasterChef.sol#1166)
Reentrancy in MasterChef.withdraw(uint256,uint256) (MasterChef.sol#1099-1119):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (MasterChef.sol#1115)
        State variables written after the call(s):
        - user.rewardDebt = user.amount.mul(pool.accCakePerShare).div(1e12) (MasterChef.sol#1117)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
ERC20.allowance(address,address).owner (MasterChef.sol#662) shadows:
        - Ownable.owner() (MasterChef.sol#540-542) (function)
ERC20._approve(address,address,uint256).owner (MasterChef.sol#834) shadows:
        - Ownable.owner() (MasterChef.sol#540-542) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
MasterChef.dev(address)._devaddr (MasterChef.sol#1200) lacks a zero-check on :
                - devaddr = _devaddr (MasterChef.sol#1202)
MasterChef.setReserveaddr(address)._addr (MasterChef.sol#1206) lacks a zero-check on :
                - reserveaddr = _addr (MasterChef.sol#1208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in MasterChef.deposit(uint256,uint256) (MasterChef.sol#1071-1096):
        External calls:
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (MasterChef.sol#1087-1091)
        Event emitted after the call(s):
        - Deposit(msg.sender,_pid,_amount) (MasterChef.sol#1095)
Reentrancy in MasterChef.emergencyWithdraw(uint256) (MasterChef.sol#1173-1180):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),user.amount) (MasterChef.sol#1176)
        Event emitted after the call(s):
        - EmergencyWithdraw(msg.sender,_pid,user.amount) (MasterChef.sol#1177)
Reentrancy in MasterChef.enterStaking(uint256) (MasterChef.sol#1122-1147):
        External calls:
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (MasterChef.sol#1136-1140)
        Event emitted after the call(s):
        - Deposit(msg.sender,0,_amount) (MasterChef.sol#1146)
Reentrancy in MasterChef.leaveStaking(uint256) (MasterChef.sol#1150-1170):
        External calls:
```

```
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (MasterChef.sol#1164)
        Event emitted after the call(s):
        - Withdraw(msg.sender,0,_amount) (MasterChef.sol#1169)
Reentrancy in MasterChef.withdraw(uint256,uint256) (MasterChef.sol#1099-1119):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (MasterChef.sol#1115)
        Event emitted after the call(s):
        - Withdraw(msg.sender,_pid,_amount) (MasterChef.sol#1118)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
MasterChef.add(uint256,IERC20,bool) (MasterChef.sol#942-962) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > startTime (MasterChef.sol#950-951)
MasterChef.set(uint256,uint256,bool) (MasterChef.sol#965-981) uses timestamp for comparisons
        Dangerous comparisons:
        - prevAllocPoint != _allocPoint (MasterChef.sol#978)
MasterChef.updateStakingPool() (MasterChef.sol#983-996) uses timestamp for comparisons
```


Copyright © 2021
Hash0X
(audit@hash0x.com)

```
        Dangerous comparisons:
        - pid < length (MasterChef.sol#986)
        - points != 0 (MasterChef.sol#989)
MasterChef.pendingCake(uint256,address) (MasterChef.sol#1008-1029) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > pool.lastRewardTime && lpSupply != 0 (MasterChef.sol#1017)
MasterChef.massUpdatePools() (MasterChef.sol#1032-1037) uses timestamp for comparisons
        Dangerous comparisons:
        - pid < length (MasterChef.sol#1034)
MasterChef.updatePool(uint256) (MasterChef.sol#1040-1068) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp <= pool.lastRewardTime (MasterChef.sol#1042)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (MasterChef.sol#293-304) uses assembly
        - INLINE ASM (MasterChef.sol#300-302)
Address._functionCallWithValue(address,bytes,uint256,string) (MasterChef.sol#401-427) uses assembly
        - INLINE ASM (MasterChef.sol#419-422)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.functionCall(address,bytes) (MasterChef.sol#348-350) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (MasterChef.sol#377-383) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (MasterChef.sol#391-399) is never used and should be removed
Address.sendValue(address,uint256) (MasterChef.sol#322-328) is never used and should be removed
Context._msgData() (MasterChef.sol#521-524) is never used and should be removed
ERC20._burn(address,uint256) (MasterChef.sol#812-818) is never used and should be removed
ERC20._burnFrom(address,uint256) (MasterChef.sol#851-858) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (MasterChef.sol#458-472) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (MasterChef.sol#483-493) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (MasterChef.sol#474-481) is never used and should be removed
SafeMath.min(uint256,uint256) (MasterChef.sol#159-161) is never used and should be removed
SafeMath.mod(uint256,uint256) (MasterChef.sol#134-136) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (MasterChef.sol#150-157) is never used and should be removed
SafeMath.sqrt(uint256) (MasterChef.sol#164-175) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (MasterChef.sol#322-328):
        - (success) = recipient.call{value: amount}() (MasterChef.sol#326)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (MasterChef.sol#401-427):
        - (success,returndata) = target.call{value: weiValue}(data) (MasterChef.sol#410)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter MasterChef.add(uint256,IERC20,bool)._allocPoint (MasterChef.sol#943) is not in mixedCase
Parameter MasterChef.add(uint256,IERC20,bool)._lpToken (MasterChef.sol#944) is not in mixedCase
Parameter MasterChef.add(uint256,IERC20,bool)._withUpdate (MasterChef.sol#945) is not in mixedCase
Parameter MasterChef.set(uint256,uint256,bool)._pid (MasterChef.sol#966) is not in mixedCase
Parameter MasterChef.set(uint256,uint256,bool)._allocPoint (MasterChef.sol#967) is not in mixedCase
Parameter MasterChef.set(uint256,uint256,bool)._withUpdate (MasterChef.sol#968) is not in mixedCase
Parameter MasterChef.getMultiplier(uint256,uint256)._from (MasterChef.sol#999) is not in mixedCase
Parameter MasterChef.getMultiplier(uint256,uint256)._to (MasterChef.sol#999) is not in mixedCase
Parameter MasterChef.pendingCake(uint256,address)._pid (MasterChef.sol#1008) is not in mixedCase
Parameter MasterChef.pendingCake(uint256,address)._user (MasterChef.sol#1008) is not in mixedCase
Parameter MasterChef.updatePool(uint256)._pid (MasterChef.sol#1040) is not in mixedCase
Parameter MasterChef.deposit(uint256,uint256)._pid (MasterChef.sol#1071) is not in mixedCase
Parameter MasterChef.deposit(uint256,uint256)._amount (MasterChef.sol#1071) is not in mixedCase
Parameter MasterChef.withdraw(uint256,uint256)._pid (MasterChef.sol#1099) is not in mixedCase
Parameter MasterChef.withdraw(uint256,uint256)._amount (MasterChef.sol#1099) is not in mixedCase
Parameter MasterChef.enterStaking(uint256)._amount (MasterChef.sol#1122) is not in mixedCase
Parameter MasterChef.leaveStaking(uint256)._amount (MasterChef.sol#1150) is not in mixedCase
Parameter MasterChef.emergencyWithdraw(uint256)._pid (MasterChef.sol#1173) is not in mixedCase
Parameter MasterChef.setCakePerSecond(uint256)._cakePerSecond (MasterChef.sol#1189) is not in mixedCase
Parameter MasterChef.dev(address)._devaddr (MasterChef.sol#1200) is not in mixedCase
Parameter MasterChef.setReserveaddr(address)._addr (MasterChef.sol#1206) is not in mixedCase
Parameter MasterChef.setHAKUPoolWeight(uint256)._weight (MasterChef.sol#1212) is not in mixedCase
Constant MasterChef.maxCakePerSecond (MasterChef.sol#905) is not in UPPER_CASE_WITH_UNDERSCORES
Variable MasterChef.BONUS_MULTIPLIER (MasterChef.sol#908) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (MasterChef.sol#522)" inContext (MasterChef.sol#516-525)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
MasterChef.burnaddr (MasterChef.sol#896) should be constant
MasterChef.startTime (MasterChef.sol#917) should be constant
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (MasterChef.sol#559-562)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (MasterChef.sol#568-572)
name() should be declared external:
        - ERC20.name() (MasterChef.sol#614-616)
decimals() should be declared external:
        - ERC20.decimals() (MasterChef.sol#621-623)
symbol() should be declared external:
        - ERC20.symbol() (MasterChef.sol#628-630)
totalSupply() should be declared external:
        - ERC20.totalSupply() (MasterChef.sol#635-637)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (MasterChef.sol#642-644)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (MasterChef.sol#654-657)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (MasterChef.sol#662-664)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (MasterChef.sol#673-676)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (MasterChef.sol#690-702)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (MasterChef.sol#716-719)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (MasterChef.sol#735-742)
mint(uint256) should be declared external:
        - ERC20.mint(uint256) (MasterChef.sol#752-755)
updateMultiplier(uint256) should be declared external:
        - MasterChef.updateMultiplier(uint256) (MasterChef.sol#932-934)
add(uint256,IERC20,bool) should be declared external:
        - MasterChef.add(uint256,IERC20,bool) (MasterChef.sol#942-962)
set(uint256,uint256,bool) should be declared external:
        - MasterChef.set(uint256,uint256,bool) (MasterChef.sol#965-981)
deposit(uint256,uint256) should be declared external:
        - MasterChef.deposit(uint256,uint256) (MasterChef.sol#1071-1096)
```


Copyright © 2021
Hash0X
(audit@hash0x.com)

```
setReserveaddr(address) should be declared external:
        - MasterChef.setReserveaddr(address) (MasterChef.sol#1206-1209)
setHAKUPoolWeight(uint256) should be declared external:
        - MasterChef.setHAKUPoolWeight(uint256) (MasterChef.sol#1212-1216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MasterChef.sol analyzed (8 contracts with 75 detectors), 95 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Syrupbar.sol

```
INFO:Detectors:
HakuToken._writeCheckpoint(address,uint32,uint256,uint256) (SyrupBar.sol#989-1007) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (SyrupBar.sol#999)
HakuSwapBar._writeCheckpoint(address,uint32,uint256,uint256) (SyrupBar.sol#1227-1253) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (SyrupBar.sol#1240-1241)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
ERC20.allowance(address,address).owner (SyrupBar.sol#577) shadows:
        - Ownable.owner() (SyrupBar.sol#455-457) (function)
ERC20._approve(address,address,uint256).owner (SyrupBar.sol#749) shadows:
        - Ownable.owner() (SyrupBar.sol#455-457) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
HakuToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (SyrupBar.sol#855-896) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,CAKE::delegateBySig: signature expired) (SyrupBar.sol#894)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (SyrupBar.sol#294-305) uses assembly
        - INLINE ASM (SyrupBar.sol#301-303)
Address._functionCallWithValue(address,bytes,uint256,string) (SyrupBar.sol#402-428) uses assembly
        - INLINE ASM (SyrupBar.sol#420-423)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (SyrupBar.sol#402-428) is never used and should be removed
Address.functionCall(address,bytes) (SyrupBar.sol#349-351) is never used and should be removed
Address.functionCall(address,bytes,string) (SyrupBar.sol#359-365) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (SyrupBar.sol#378-384) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (SyrupBar.sol#392-400) is never used and should be removed
Address.isContract(address) (SyrupBar.sol#294-305) is never used and should be removed
Address.sendValue(address,uint256) (SyrupBar.sol#323-329) is never used and should be removed
Context._msgData() (SyrupBar.sol#426-439) is never used and should be removed
ERC20._burnFrom(address,uint256) (SyrupBar.sol#766-773) is never used and should be removed
HakuSwapBar._writeCheckpoint(address,uint32,uint256,uint256) (SyrupBar.sol#1227-1253) is never used and should be removed
HakuSwapBar.safe32(uint256,string) (SyrupBar.sol#1255-1262) is never used and should be removed
SafeMath.div(uint256,uint256) (SyrupBar.sol#95-97) is never used and should be removed
SafeMath.div(uint256,uint256,string) (SyrupBar.sol#111-121) is never used and should be removed
SafeMath.min(uint256,uint256) (SyrupBar.sol#160-162) is never used and should be removed
```

```
SafeMath.mod(uint256,uint256) (SyrupBar.sol#135-137) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (SyrupBar.sol#151-158) is never used and should be removed
SafeMath.mul(uint256,uint256) (SyrupBar.sol#69-81) is never used and should be removed
SafeMath.sqrt(uint256) (SyrupBar.sol#165-176) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (SyrupBar.sol#323-329):
        - (success) = recipient.call{value: amount}() (SyrupBar.sol#327)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (SyrupBar.sol#402-428):
        - (success,returndata) = target.call{value: weiValue}(data) (SyrupBar.sol#411)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter HakuToken.mintFor(address,uint256)._to (SyrupBar.sol#780) is not in mixedCase
Parameter HakuToken.mintFor(address,uint256)._amount (SyrupBar.sol#780) is not in mixedCase
Constant HakuToken.maxSupply (SyrupBar.sol#778) is not in UPPER_CASE_WITH_UNDERSCORES
Variable HakuToken._delegates (SyrupBar.sol#798) is not in mixedCase
Parameter HakuSwapBar.mint(address,uint256)._to (SyrupBar.sol#1022) is not in mixedCase
Parameter HakuSwapBar.mint(address,uint256)._amount (SyrupBar.sol#1022) is not in mixedCase
Parameter HakuSwapBar.burn(address,uint256)._from (SyrupBar.sol#1027) is not in mixedCase
Parameter HakuSwapBar.burn(address,uint256)._amount (SyrupBar.sol#1027) is not in mixedCase
Variable HakuSwapBar._delegates (SyrupBar.sol#1050) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (SyrupBar.sol#437)" inContext (SyrupBar.sol#431-440)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (SyrupBar.sol#474-477)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (SyrupBar.sol#483-487)
decimals() should be declared external:
        - ERC20.decimals() (SyrupBar.sol#536-538)
symbol() should be declared external:
        - ERC20.symbol() (SyrupBar.sol#543-545)
transfer(address,uint256) should be declared external:
```

```
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (SyrupBar.sol#577-579)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (SyrupBar.sol#588-591)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (SyrupBar.sol#605-617)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (SyrupBar.sol#631-634)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (SyrupBar.sol#650-657)
mint(uint256) should be declared external:
        - ERC20.mint(uint256) (SyrupBar.sol#667-670)
        - HakuToken.mint(uint256) (SyrupBar.sol#786-790)
mintFor(address,uint256) should be declared external:
        - HakuToken.mintFor(address,uint256) (SyrupBar.sol#780-784)
mint(address,uint256) should be declared external:
        - HakuSwapBar.mint(address,uint256) (SyrupBar.sol#1022-1025)
burn(address,uint256) should be declared external:
        - HakuSwapBar.burn(address,uint256) (SyrupBar.sol#1027-1030)
safeCakeTransfer(address,uint256) should be declared external:
        - HakuSwapBar.safeCakeTransfer(address,uint256) (SyrupBar.sol#1038-1041)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SyrupBar.sol analyzed (8 contracts with 75 detectors), 53 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```


Copyright © 2021
Hash0X
(audit@hash0x.com)

Slither log >> Factory.sol

```
INFO:Detectors:
HakuswapFactory.constructor(address)._feeToSetter (Factory.sol#18) lacks a zero-check on :
                - feeToSetter = _feeToSetter (Factory.sol#19)
HakuswapFactory.setFeeTo(address)._feeTo (Factory.sol#41) lacks a zero-check on :
                - feeTo = _feeTo (Factory.sol#43)
HakuswapFactory.setFeeToSetter(address)._feeToSetter (Factory.sol#46) lacks a zero-check on :
                - feeToSetter = _feeToSetter (Factory.sol#48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Parameter HakuswapFactory.setFeeTo(address)._feeTo (Factory.sol#41) is not in mixedCase
Parameter HakuswapFactory.setFeeToSetter(address)._feeToSetter (Factory.sol#46) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Slither:Factory.sol analyzed (1 contracts with 75 detectors), 5 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Router.sol

```
INFO:Detectors:
HakuswapRouter.removeLiquidity(address,address,uint256,uint256,uint256,address,uint256) (Router.sol#437-453) ignores return value by IHak
uswapPair(pair).transferFrom(msg.sender,pair,liquidity) (Router.sol#447)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
HakuswapLibrary.getAmountsOut(address,uint256,address[]).i (Router.sol#306) is a local variable never initialized
HakuswapRouter._swap(uint256[],address[],address).i (Router.sol#547) is a local variable never initialized
HakuswapRouter._swapSupportingFeeOnTransferTokens(address[],address).i (Router.sol#656) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
HakuswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256) (Router.sol#367-394) ignores return value by IHakuswapFacto
ry(factory).createPair(tokenA,tokenB) (Router.sol#377)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
HakuswapRouter.constructor(address,address)._factory (Router.sol#357) lacks a zero-check on :
                - factory = _factory (Router.sol#358)
HakuswapRouter.constructor(address,address)._WETH (Router.sol#357) lacks a zero-check on :
                - WETH = _WETH (Router.sol#359)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
HakuswapRouter._swap(uint256[],address[],address) (Router.sol#546-557) has external calls inside a loop: IHakuswapPair(HakuswapLibrary.pa
irFor(factory,input,output)).swap(amount0Out,amount1Out,to,new bytes(0)) (Router.sol#553-555)
HakuswapRouter._swapSupportingFeeOnTransferTokens(address[],address) (Router.sol#655-672) has external calls inside a loop: (reserve0,res
erve1) = pair.getReserves() (Router.sol#663)
HakuswapRouter._swapSupportingFeeOnTransferTokens(address[],address) (Router.sol#655-672) has external calls inside a loop: amountInput =
 IERC20(input).balanceOf(address(pair)).sub(reserveInput) (Router.sol#665)
HakuswapRouter._swapSupportingFeeOnTransferTokens(address[],address) (Router.sol#655-672) has external calls inside a loop: pair.swap(amo
unt0Out,amount1Out,to,new bytes(0)) (Router.sol#670)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
TransferHelper.safeApprove(address,address,uint256) (Router.sol#21-25) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in TransferHelper.safeApprove(address,address,uint256) (Router.sol#21-25):
        - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (Router.sol#23)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (Router.sol#27-31):
```

```
        - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (Router.sol#29)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (Router.sol#33-37):
        - (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (Router.sol#35)
Low level call in TransferHelper.safeTransferETH(address,uint256) (Router.sol#39-42):
        - (success) = to.call{value: value}(new bytes(0)) (Router.sol#40)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IHakuswapRouter01.WETH() (Router.sol#47) is not in mixedCase
Function IHakuswapPair.DOMAIN_SEPARATOR() (Router.sol#195) is not in mixedCase
Function IHakuswapPair.PERMIT_TYPEHASH() (Router.sol#196) is not in mixedCase
Function IHakuswapPair.MINIMUM_LIQUIDITY() (Router.sol#213) is not in mixedCase
Variable HakuswapRouter.WETH (Router.sol#350) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable IHakuswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Router.sol#52) i
s too similar to IHakuswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (Router.s
ol#53)
Variable IHakuswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Router.sol#52) i
s too similar to HakuswapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (Router.sol#
399)
Variable HakuswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (Router.sol#370) is too similar to
HakuswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (Router.sol#371)
Variable HakuswapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Router.sol#398) is
too similar to HakuswapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (Router.sol#39
9)
Variable HakuswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (Router.sol#370) is too similar to
HakuswapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (Router.sol#399)
Variable IHakuswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Router.sol#52) i
s too similar to HakuswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (Router.sol#371)
Variable HakuswapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Router.sol#398) is
too similar to IHakuswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (Router.sol
#53)
Variable HakuswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (Router.sol#370) is too similar to
IHakuswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (Router.sol#53)
Variable HakuswapRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Router.sol#398) is
too similar to HakuswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (Router.sol#371)
Variable HakuswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountAOptimal (Router.sol#388) is too similar to
HakuswapRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBOptimal (Router.sol#383)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
quote(uint256,uint256,uint256) should be declared external:
        - HakuswapRouter.quote(uint256,uint256,uint256) (Router.sol#737-739)
getAmountOut(uint256,uint256,uint256) should be declared external:
        - HakuswapRouter.getAmountOut(uint256,uint256,uint256) (Router.sol#741-749)
getAmountIn(uint256,uint256,uint256) should be declared external:
        - HakuswapRouter.getAmountIn(uint256,uint256,uint256) (Router.sol#751-759)
getAmountsOut(uint256,address[]) should be declared external:
        - HakuswapRouter.getAmountsOut(uint256,address[]) (Router.sol#761-769)
getAmountsIn(uint256,address[]) should be declared external:
        - HakuswapRouter.getAmountsIn(uint256,address[]) (Router.sol#771-779)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Router.sol analyzed (10 contracts with 75 detectors), 36 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```


Copyright © 2021
Hash0X
(audit@hash0x.com)

Slither log >> HakuToken.sol

```
INFO:Detectors:
HakuToken._writeCheckpoint(address,uint32,uint256,uint256) (HakuToken.sol#1063-1081) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (HakuToken.sol#1073)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
ERC20.allowance(address,address).owner (HakuToken.sol#652) shadows:
        - Ownable.owner() (HakuToken.sol#544-546) (function)
ERC20._approve(address,address,uint256).owner (HakuToken.sol#824) shadows:
        - Ownable.owner() (HakuToken.sol#544-546) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
HakuToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (HakuToken.sol#929-970) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,CAKE::delegateBySig: signature expired) (HakuToken.sol#968)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (HakuToken.sol#293-304) uses assembly
        - INLINE ASM (HakuToken.sol#300-302)
Address._functionCallWithValue(address,bytes,uint256,string) (HakuToken.sol#401-427) uses assembly
        - INLINE ASM (HakuToken.sol#419-422)
HakuToken.getChainId() (HakuToken.sol#1088-1092) uses assembly
        - INLINE ASM (HakuToken.sol#1090)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (HakuToken.sol#401-427) is never used and should be removed
Address.functionCall(address,bytes) (HakuToken.sol#348-350) is never used and should be removed
Address.functionCall(address,bytes,string) (HakuToken.sol#358-364) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (HakuToken.sol#377-383) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (HakuToken.sol#391-399) is never used and should be removed
Address.isContract(address) (HakuToken.sol#293-304) is never used and should be removed
Address.sendValue(address,uint256) (HakuToken.sol#322-328) is never used and should be removed
Context._msgData() (HakuToken.sol#521-524) is never used and should be removed
ERC20._burn(address,uint256) (HakuToken.sol#802-808) is never used and should be removed
ERC20._burnFrom(address,uint256) (HakuToken.sol#841-848) is never used and should be removed
SafeERC20._callOptionalReturn(IERC20,bytes) (HakuToken.sol#501-512) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (HakuToken.sol#458-472) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (HakuToken.sol#483-493) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (HakuToken.sol#474-481) is never used and should be removed
```

```
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (HakuToken.sol#474-481) is never used and should be removed
SafeERC20.safeTransfer(IERC20,address,uint256) (HakuToken.sol#434-440) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (HakuToken.sol#442-449) is never used and should be removed
SafeMath.div(uint256,uint256) (HakuToken.sol#94-96) is never used and should be removed
SafeMath.div(uint256,uint256,string) (HakuToken.sol#110-120) is never used and should be removed
SafeMath.min(uint256,uint256) (HakuToken.sol#159-161) is never used and should be removed
SafeMath.mod(uint256,uint256) (HakuToken.sol#134-136) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (HakuToken.sol#150-157) is never used and should be removed
SafeMath.mul(uint256,uint256) (HakuToken.sol#68-80) is never used and should be removed
SafeMath.sqrt(uint256) (HakuToken.sol#164-175) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.4.0 (HakuToken.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (HakuToken.sol#322-328):
        - (success) = recipient.call{value: amount}() (HakuToken.sol#326)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (HakuToken.sol#401-427):
        - (success,returndata) = target.call{value: weiValue}(data) (HakuToken.sol#410)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter HakuToken.mintFor(address,uint256)._to (HakuToken.sol#854) is not in mixedCase
Parameter HakuToken.mintFor(address,uint256)._amount (HakuToken.sol#854) is not in mixedCase
Constant HakuToken.maxSupply (HakuToken.sol#852) is not in UPPER_CASE_WITH_UNDERSCORES
Variable HakuToken._delegates (HakuToken.sol#872) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (HakuToken.sol#522)" inContext (HakuToken.sol#516-525)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (HakuToken.sol#563-566)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (HakuToken.sol#572-576)
decimals() should be declared external:
        - ERC20.decimals() (HakuToken.sol#611-613)
symbol() should be declared external:
        - ERC20.symbol() (HakuToken.sol#618-620)
```

```
symbol() should be declared external:
        - ERC20.symbol() (HakuToken.sol#618-620)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (HakuToken.sol#644-647)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (HakuToken.sol#652-654)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (HakuToken.sol#663-666)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (HakuToken.sol#680-692)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (HakuToken.sol#706-709)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (HakuToken.sol#725-732)
mint(uint256) should be declared external:
        - ERC20.mint(uint256) (HakuToken.sol#742-745)
        - HakuToken.mint(uint256) (HakuToken.sol#860-864)
mintFor(address,uint256) should be declared external:
        - HakuToken.mintFor(address,uint256) (HakuToken.sol#854-858)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:HakuToken.sol analyzed (8 contracts with 75 detectors), 50 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```


Copyright © 2021
Hash0X
(audit@hash0x.com)

Slither log >> xHAKU.sol

```
INFO:Detectors:
HakuStakingToken.haku (xHAKU.sol#854) is never initialized. It is used in:
        - HakuStakingToken.enter(uint256) (xHAKU.sol#860-876)
        - HakuStakingToken.leave(uint256) (xHAKU.sol#879-886)
        - HakuStakingToken.HAKUBalance(address) (xHAKU.sol#889-893)
        - HakuStakingToken.xHAKUForHAKU(uint256) (xHAKU.sol#896-899)
        - HakuStakingToken.HAKUForxHAKU(uint256) (xHAKU.sol#902-911)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
HakuStakingToken.HAKUForxHAKU(uint256) (xHAKU.sol#902-911) uses a dangerous strict equality:
        - totalxHAKU == 0 || totalHaku == 0 (xHAKU.sol#905)
HakuStakingToken._writeCheckpoint(address,uint32,uint256,uint256) (xHAKU.sol#1146-1164) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (xHAKU.sol#1156)
HakuStakingToken.enter(uint256) (xHAKU.sol#860-876) uses a dangerous strict equality:
        - totalShares == 0 || totalHaku == 0 (xHAKU.sol#866)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
ERC20.allowance(address,address).owner (xHAKU.sol#666) shadows:
        - Ownable.owner() (xHAKU.sol#544-546) (function)
ERC20._approve(address,address,uint256).owner (xHAKU.sol#825) shadows:
        - Ownable.owner() (xHAKU.sol#544-546) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
HakuStakingToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (xHAKU.sol#1021-1053) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,HAKU::delegateBySig: signature expired) (xHAKU.sol#1051)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (xHAKU.sol#293-304) uses assembly
        - INLINE ASM (xHAKU.sol#300-302)
Address._functionCallWithValue(address,bytes,uint256,string) (xHAKU.sol#401-427) uses assembly
        - INLINE ASM (xHAKU.sol#419-422)
HakuStakingToken.getChainId() (xHAKU.sol#1171-1175) uses assembly
        - INLINE ASM (xHAKU.sol#1173)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (xHAKU.sol#401-427) is never used and should be removed
Address.functionCall(address,bytes) (xHAKU.sol#348-350) is never used and should be removed
```

```
Address.functionCall(address,bytes,string) (xHAKU.sol#358-364) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (xHAKU.sol#377-383) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (xHAKU.sol#391-399) is never used and should be removed
Address.isContract(address) (xHAKU.sol#293-304) is never used and should be removed
Address.sendValue(address,uint256) (xHAKU.sol#322-328) is never used and should be removed
Context._msgData() (xHAKU.sol#521-524) is never used and should be removed
ERC20._burnFrom(address,uint256) (xHAKU.sol#842-849) is never used and should be removed
ERC20._transfer(address,address,uint256) (xHAKU.sol#762-773) is never used and should be removed
SafeERC20._callOptionalReturn(IERC20,bytes) (xHAKU.sol#501-512) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (xHAKU.sol#458-472) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (xHAKU.sol#483-493) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (xHAKU.sol#474-481) is never used and should be removed
SafeERC20.safeTransfer(IERC20,address,uint256) (xHAKU.sol#434-440) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (xHAKU.sol#442-449) is never used and should be removed
SafeMath.min(uint256,uint256) (xHAKU.sol#159-161) is never used and should be removed
SafeMath.mod(uint256,uint256) (xHAKU.sol#134-136) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (xHAKU.sol#150-157) is never used and should be removed
SafeMath.sqrt(uint256) (xHAKU.sol#164-175) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>0.6.6 (xHAKU.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (xHAKU.sol#322-328):
        - (success) = recipient.call{value: amount}() (xHAKU.sol#326)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (xHAKU.sol#401-427):
        - (success,returndata) = target.call{value: weiValue}(data) (xHAKU.sol#410)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter HakuStakingToken.enter(uint256)._amount (xHAKU.sol#860) is not in mixedCase
Parameter HakuStakingToken.leave(uint256)._share (xHAKU.sol#879) is not in mixedCase
Function HakuStakingToken.HAKUBalance(address) (xHAKU.sol#889-893) is not in mixedCase
Parameter HakuStakingToken.HAKUBalance(address)._account (xHAKU.sol#889) is not in mixedCase
Parameter HakuStakingToken.xHAKUForHAKU(uint256)._xHAKUAmount (xHAKU.sol#896) is not in mixedCase
Function HakuStakingToken.HAKUForxHAKU(uint256) (xHAKU.sol#902-911) is not in mixedCase
Parameter HakuStakingToken.HAKUForxHAKU(uint256)._hakuAmount (xHAKU.sol#902) is not in mixedCase
Parameter HakuStakingToken.burn(address,uint256)._from (xHAKU.sol#949) is not in mixedCase
Parameter HakuStakingToken.burn(address,uint256)._amount (xHAKU.sol#949) is not in mixedCase
```

```
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (xHAKU.sol#563-566)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (xHAKU.sol#572-576)
decimals() should be declared external:
        - ERC20.decimals() (xHAKU.sol#625-627)
symbol() should be declared external:
        - ERC20.symbol() (xHAKU.sol#632-634)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (xHAKU.sol#666-668)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (xHAKU.sol#677-680)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (xHAKU.sol#707-710)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (xHAKU.sol#726-733)
mint(uint256) should be declared external:
        - ERC20.mint(uint256) (xHAKU.sol#743-746)
enter(uint256) should be declared external:
        - HakuStakingToken.enter(uint256) (xHAKU.sol#860-876)
leave(uint256) should be declared external:
        - HakuStakingToken.leave(uint256) (xHAKU.sol#879-886)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:xHAKU.sol analyzed (8 contracts with 75 detectors), 62 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```


Copyright © 2021
Hash0X
(audit@hash0x.com)

Slither log >> LakeOfHaku.sol

```
INFO:Detectors:
LakeOfHaku._convert(address,address) (LakeOfHaku.sol#827-866) ignores return value by pair.burn(address(this)) (LakeOfHaku.sol#850)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
LakeOfHaku.constructor(address,address,address,address)._xhaku (LakeOfHaku.sol#733) lacks a zero-check on :
                - xhaku = _xhaku (LakeOfHaku.sol#738)
LakeOfHaku.constructor(address,address,address,address)._haku (LakeOfHaku.sol#734) lacks a zero-check on :
                - haku = _haku (LakeOfHaku.sol#739)
LakeOfHaku.constructor(address,address,address,address)._wcro (LakeOfHaku.sol#735) lacks a zero-check on :
                - wcro = _wcro (LakeOfHaku.sol#740)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in LakeOfHaku._convert(address,address) (LakeOfHaku.sol#827-866):
        External calls:
        - IERC20(address(pair)).safeTransfer(address(pair),pair.balanceOf(address(this))) (LakeOfHaku.sol#841-844)
        - pair.burn(address(this)) (LakeOfHaku.sol#850)
        - LogConvert(msg.sender,token0,token1,amount0,amount1,_convertStep(token0,token1,amount0,amount1)) (LakeOfHaku.sol#858-865)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (LakeOfHaku.sol#506)
                - IERC20(token).safeTransfer(devAddr,amount) (LakeOfHaku.sol#975)
                - (success,returndata) = target.call{value: weiValue}(data) (LakeOfHaku.sol#410)
                - IERC20(haku).safeTransfer(xhaku,amount) (LakeOfHaku.sol#881)
                - IERC20(fromToken).safeTransfer(address(pair),amountIn) (LakeOfHaku.sol#958)
                - IERC20(haku).safeTransfer(xhaku,amount0) (LakeOfHaku.sol#892)
                - pair.swap(amount0Out,amount1Out,to,new bytes(0)) (LakeOfHaku.sol#963)
                - IERC20(haku).safeTransfer(xhaku,amount1) (LakeOfHaku.sol#896)
        External calls sending eth:
        - LogConvert(msg.sender,token0,token1,amount0,amount1,_convertStep(token0,token1,amount0,amount1)) (LakeOfHaku.sol#858-865)
                - (success,returndata) = target.call{value: weiValue}(data) (LakeOfHaku.sol#410)
        Event emitted after the call(s):
        - LogConvert(msg.sender,token0,token1,amount0,amount1,_convertStep(token0,token1,amount0,amount1)) (LakeOfHaku.sol#858-865)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.isContract(address) (LakeOfHaku.sol#293-304) uses assembly
        - INLINE ASM (LakeOfHaku.sol#300-302)
Address._functionCallWithValue(address,bytes,uint256,string) (LakeOfHaku.sol#401-427) uses assembly
        - INLINE ASM (LakeOfHaku.sol#419-422)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
```

```
INFO:Detectors:
Address.functionCall(address,bytes) (LakeOfHaku.sol#348-350) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (LakeOfHaku.sol#377-383) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (LakeOfHaku.sol#391-399) is never used and should be removed
Address.sendValue(address,uint256) (LakeOfHaku.sol#322-328) is never used and should be removed
Context._msgData() (LakeOfHaku.sol#530-533) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (LakeOfHaku.sol#458-472) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (LakeOfHaku.sol#483-493) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (LakeOfHaku.sol#474-481) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (LakeOfHaku.sol#442-449) is never used and should be removed
SafeMath.min(uint256,uint256) (LakeOfHaku.sol#159-161) is never used and should be removed
SafeMath.mod(uint256,uint256) (LakeOfHaku.sol#134-136) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (LakeOfHaku.sol#150-157) is never used and should be removed
SafeMath.sqrt(uint256) (LakeOfHaku.sol#164-175) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (LakeOfHaku.sol#322-328):
        - (success) = recipient.call{value: amount}() (LakeOfHaku.sol#326)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (LakeOfHaku.sol#401-427):
        - (success,returndata) = target.call{value: weiValue}(data) (LakeOfHaku.sol#410)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IUniswapV2ERC20.DOMAIN_SEPARATOR() (LakeOfHaku.sol#616) is not in mixedCase
Function IUniswapV2ERC20.PERMIT_TYPEHASH() (LakeOfHaku.sol#617) is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (LakeOfHaku.sol#638) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (LakeOfHaku.sol#639) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (LakeOfHaku.sol#656) is not in mixedCase
Parameter LakeOfHaku.addAuth(address)._auth (LakeOfHaku.sol#746) is not in mixedCase
Parameter LakeOfHaku.revokeAuth(address)._auth (LakeOfHaku.sol#751) is not in mixedCase
Parameter LakeOfHaku.setDevCut(uint256)._amount (LakeOfHaku.sol#772) is not in mixedCase
Parameter LakeOfHaku.setDevAddr(address)._addr (LakeOfHaku.sol#779) is not in mixedCase
Variable LakeOfHaku._bridges (LakeOfHaku.sol#717) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (LakeOfHaku.sol#531)" inContext (LakeOfHaku.sol#525-534)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
owner() should be declared external:
```

```
INFO:Detectors:
owner() should be declared external:
        - Ownable.owner() (LakeOfHaku.sol#565-567)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (LakeOfHaku.sol#584-587)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (LakeOfHaku.sol#593-597)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:LakeOfHaku.sol analyzed (10 contracts with 75 detectors), 36 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```


Copyright © 2021
Hash0X
(audit@hash0x.com)

Slither log >> Multicall.sol

```
INFO:Detectors:
Multicall.aggregate(Multicall.Call[]) (Multicall.sol#12-20) has external calls inside a loop: (success,ret) = calls[i].target.call(calls[
i].callData) (Multicall.sol#16)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Low level call in Multicall.aggregate(Multicall.Call[]) (Multicall.sol#12-20):
        - (success,ret) = calls[i].target.call(calls[i].callData) (Multicall.sol#16)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
aggregate(Multicall.Call[]) should be declared external:
        - Multicall.aggregate(Multicall.Call[]) (Multicall.sol#12-20)
getEthBalance(address) should be declared external:
        - Multicall.getEthBalance(address) (Multicall.sol#22-24)
getBlockHash(uint256) should be declared external:
        - Multicall.getBlockHash(uint256) (Multicall.sol#25-27)
getLastBlockHash() should be declared external:
        - Multicall.getLastBlockHash() (Multicall.sol#28-30)
getCurrentBlockTimestamp() should be declared external:
        - Multicall.getCurrentBlockTimestamp() (Multicall.sol#31-33)
getCurrentBlockDifficulty() should be declared external:
        - Multicall.getCurrentBlockDifficulty() (Multicall.sol#34-36)
getCurrentBlockGasLimit() should be declared external:
        - Multicall.getCurrentBlockGasLimit() (Multicall.sol#37-39)
getCurrentBlockCoinbase() should be declared external:
        - Multicall.getCurrentBlockCoinbase() (Multicall.sol#40-42)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Multicall.sol analyzed (1 contracts with 75 detectors), 10 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> Oracle.sol

```
INFO:Detectors:
Oracle.constructor(address).factory_ (Oracle.sol#426) lacks a zero-check on :
                - factory = factory_ (Oracle.sol#427)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
HakuswapOracleLibrary.currentCumulativePrices(address) (Oracle.sol#389-407) uses timestamp for comparisons
        Dangerous comparisons:
        - blockTimestampLast != blockTimestamp (Oracle.sol#398)
Oracle.update(address,address) (Oracle.sol#431-444) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(timeElapsed >= CYCLE,MDEXOracle: PERIOD_NOT_ELAPSED) (Oracle.sol#439)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
FixedPoint.decode(FixedPoint.uq112x112) (Oracle.sol#370-372) is never used and should be removed
FixedPoint.div(FixedPoint.uq112x112,uint112) (Oracle.sol#349-352) is never used and should be removed
FixedPoint.encode(uint112) (Oracle.sol#339-341) is never used and should be removed
FixedPoint.encode144(uint144) (Oracle.sol#344-346) is never used and should be removed
HakuswapLibrary.getAmountIn(uint256,uint256,uint256) (Oracle.sol#224-230) is never used and should be removed
HakuswapLibrary.getAmountOut(uint256,uint256,uint256) (Oracle.sol#214-221) is never used and should be removed
HakuswapLibrary.getAmountsIn(address,uint256,address[]) (Oracle.sol#244-252) is never used and should be removed
HakuswapLibrary.getAmountsOut(address,uint256,address[]) (Oracle.sol#233-241) is never used and should be removed
HakuswapLibrary.getReserves(address,address,address) (Oracle.sol#199-204) is never used and should be removed
HakuswapLibrary.pairFor(address,address,address) (Oracle.sol#188-196) is never used and should be removed
HakuswapLibrary.quote(uint256,uint256,uint256) (Oracle.sol#207-211) is never used and should be removed
SafeMath.add(uint256,uint256) (Oracle.sol#16-21) is never used and should be removed
SafeMath.div(uint256,uint256) (Oracle.sol#94-96) is never used and should be removed
SafeMath.div(uint256,uint256,string) (Oracle.sol#110-120) is never used and should be removed
SafeMath.min(uint256,uint256) (Oracle.sol#159-161) is never used and should be removed
SafeMath.mod(uint256,uint256) (Oracle.sol#134-136) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (Oracle.sol#150-157) is never used and should be removed
SafeMath.mul(uint256,uint256) (Oracle.sol#68-80) is never used and should be removed
SafeMath.sqrt(uint256) (Oracle.sol#164-175) is never used and should be removed
SafeMath.sub(uint256,uint256) (Oracle.sol#33-35) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (Oracle.sol#47-56) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.6.6 (Oracle.sol#3) allows old versions
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function IHakuswapPair.DOMAIN_SEPARATOR() (Oracle.sol#271) is not in mixedCase
Function IHakuswapPair.PERMIT_TYPEHASH() (Oracle.sol#272) is not in mixedCase
Function IHakuswapPair.MINIMUM_LIQUIDITY() (Oracle.sol#289) is not in mixedCase
Struct FixedPoint.uq112x112 (Oracle.sol#326-328) is not in CapWords
Struct FixedPoint.uq144x112 (Oracle.sol#332-334) is not in CapWords
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable HakuswapOracleLibrary.currentCumulativePrices(address).price0Cumulative (Oracle.sol#391) is too similar to HakuswapOracleLibrary
.currentCumulativePrices(address).price1Cumulative (Oracle.sol#391)
Variable Oracle.consult(address,uint256,address).price0Cumulative (Oracle.sol#463) is too similar to Oracle.consult(address,uint256,addre
ss).price1Cumulative (Oracle.sol#463)
Variable Oracle.update(address,address).price0Cumulative (Oracle.sol#440) is too similar to Oracle.update(address,address).price1Cumulati
ve (Oracle.sol#440)
Variable Oracle.consult(address,uint256,address).price0Cumulative (Oracle.sol#463) is too similar to Oracle.update(address,address).price
1Cumulative (Oracle.sol#440)
Variable Oracle.update(address,address).price0Cumulative (Oracle.sol#440) is too similar to Oracle.consult(address,uint256,address).price
1Cumulative (Oracle.sol#463)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Slither:Oracle.sol analyzed (7 contracts with 75 detectors), 38 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```


Copyright © 2021
Hash0X
(audit@hash0x.com)

Slither log >> SwapMining.sol

```
INFO:Detectors:
ERC20.allowance(address,address).owner (SwapMining.sol#1002) shadows:
        - Ownable.owner() (SwapMining.sol#880-882) (function)
ERC20._approve(address,address,uint256).owner (SwapMining.sol#1174) shadows:
        - Ownable.owner() (SwapMining.sol#880-882) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
SwapMining.constructor(HakuToken,IHakuswapFactory,IOracle,address,address,uint256,uint256)._targetToken (SwapMining.sol#1476) lacks a zer
o-check on :
                - targetToken = _targetToken (SwapMining.sol#1488)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
SwapMining.getQuantity(address,uint256,address) (SwapMining.sol#1696-1714) has external calls inside a loop: factory.getPair(outputToken,
intermediate) != address(0) && factory.getPair(intermediate,anchorToken) != address(0) (SwapMining.sol#1706)
SwapMining.getQuantity(address,uint256,address) (SwapMining.sol#1696-1714) has external calls inside a loop: interQuantity = IOracle(orac
le).consult(outputToken,outputAmount,intermediate) (SwapMining.sol#1707)
SwapMining.getQuantity(address,uint256,address) (SwapMining.sol#1696-1714) has external calls inside a loop: quantity = IOracle(oracle).c
onsult(intermediate,interQuantity,anchorToken) (SwapMining.sol#1708)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
HakuToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (SwapMining.sol#1280-1321) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,CAKE::delegateBySig: signature expired) (SwapMining.sol#1319)
SwapMining.addPair(uint256,address,bool) (SwapMining.sol#1516-1532) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > startTime (SwapMining.sol#1521)
SwapMining.phase(uint256) (SwapMining.sol#1588-1596) uses timestamp for comparisons
        Dangerous comparisons:
        - blockTimestamp > startTime (SwapMining.sol#1592)
SwapMining.getHakuReward(uint256) (SwapMining.sol#1612-1628) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_lastRewardTime <= block.timestamp,SwapMining: must little than the current block timestamp) (SwapMining.s
ol#1613)
        - n < m (SwapMining.sol#1618)
SwapMining.massMintPools() (SwapMining.sol#1631-1636) uses timestamp for comparisons
        Dangerous comparisons:
        - pid < length (SwapMining.sol#1633)
SwapMining.mint(uint256) (SwapMining.sol#1638-1653) uses timestamp for comparisons
```

```
        Dangerous comparisons:
        - block.timestamp <= pool.lastRewardTime (SwapMining.sol#1640)
        - blockReward <= 0 (SwapMining.sol#1644)
SwapMining.swap(address,address,address,uint256) (SwapMining.sol#1661-1694) uses timestamp for comparisons
        Dangerous comparisons:
        - poolLength() <= 0 (SwapMining.sol#1666)
SwapMining.takerWithdraw() (SwapMining.sol#1717-1739) uses timestamp for comparisons
        Dangerous comparisons:
        - pid < length (SwapMining.sol#1720)
SwapMining.getUserReward(uint256,address) (SwapMining.sol#1742-1754) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_pid <= poolInfo.length - 1,SwapMining: Not find this pool) (SwapMining.sol#1743)
        - user.quantity > 0 (SwapMining.sol#1747)
SwapMining.getPoolInfo(uint256) (SwapMining.sol#1757-1768) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_pid <= poolInfo.length - 1,SwapMining: Not find this pool) (SwapMining.sol#1758)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (SwapMining.sol#710-721) uses assembly
        - INLINE ASM (SwapMining.sol#717-719)
Address._functionCallWithValue(address,bytes,uint256,string) (SwapMining.sol#818-844) uses assembly
        - INLINE ASM (SwapMining.sol#836-839)
HakuToken.getChainId() (SwapMining.sol#1439-1443) uses assembly
        - INLINE ASM (SwapMining.sol#1441)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (SwapMining.sol#818-844) is never used and should be removed
Address.functionCall(address,bytes) (SwapMining.sol#765-767) is never used and should be removed
Address.functionCall(address,bytes,string) (SwapMining.sol#775-781) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (SwapMining.sol#794-800) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (SwapMining.sol#808-816) is never used and should be removed
Address.isContract(address) (SwapMining.sol#710-721) is never used and should be removed
Address.sendValue(address,uint256) (SwapMining.sol#739-745) is never used and should be removed
Context._msgData() (SwapMining.sol#857-860) is never used and should be removed
ERC20._burn(address,uint256) (SwapMining.sol#1152-1158) is never used and should be removed
ERC20._burnFrom(address,uint256) (SwapMining.sol#1191-1198) is never used and should be removed
EnumerableSet.add(EnumerableSet.Bytes32Set,bytes32) (SwapMining.sol#540-542) is never used and should be removed
EnumerableSet.add(EnumerableSet.UintSet,uint256) (SwapMining.sol#649-651) is never used and should be removed
```

```
EnumerableSet.at(EnumerableSet.Bytes32Set,uint256) (SwapMining.sol#578-580) is never used and should be removed
EnumerableSet.at(EnumerableSet.UintSet,uint256) (SwapMining.sol#687-689) is never used and should be removed
EnumerableSet.contains(EnumerableSet.Bytes32Set,bytes32) (SwapMining.sol#557-559) is never used and should be removed
EnumerableSet.contains(EnumerableSet.UintSet,uint256) (SwapMining.sol#666-668) is never used and should be removed
EnumerableSet.length(EnumerableSet.Bytes32Set) (SwapMining.sol#564-566) is never used and should be removed
EnumerableSet.length(EnumerableSet.UintSet) (SwapMining.sol#673-675) is never used and should be removed
EnumerableSet.remove(EnumerableSet.Bytes32Set,bytes32) (SwapMining.sol#550-552) is never used and should be removed
EnumerableSet.remove(EnumerableSet.UintSet,uint256) (SwapMining.sol#659-661) is never used and should be removed
HakuswapLibrary.getAmountIn(uint256,uint256,uint256) (SwapMining.sol#53-59) is never used and should be removed
HakuswapLibrary.getAmountOut(uint256,uint256,uint256) (SwapMining.sol#43-50) is never used and should be removed
HakuswapLibrary.getAmountsIn(address,uint256,address[]) (SwapMining.sol#73-81) is never used and should be removed
HakuswapLibrary.getAmountsOut(address,uint256,address[]) (SwapMining.sol#62-70) is never used and should be removed
HakuswapLibrary.getReserves(address,address,address) (SwapMining.sol#28-33) is never used and should be removed
HakuswapLibrary.quote(uint256,uint256,uint256) (SwapMining.sol#36-40) is never used and should be removed
SafeMath.min(uint256,uint256) (SwapMining.sol#307-309) is never used and should be removed
SafeMath.mod(uint256,uint256) (SwapMining.sol#282-284) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (SwapMining.sol#298-305) is never used and should be removed
SafeMath.sqrt(uint256) (SwapMining.sol#312-323) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.6.0<0.8.0 (SwapMining.sol#3) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (SwapMining.sol#739-745):
        - (success) = recipient.call{value: amount}() (SwapMining.sol#743)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (SwapMining.sol#818-844):
        - (success,returndata) = target.call{value: weiValue}(data) (SwapMining.sol#827)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```


Copyright © 2021
Hash0X
(audit@hash0x.com)

```
INFO:Detectors:
Function IHakuswapPair.DOMAIN_SEPARATOR() (SwapMining.sol#100) is not in mixedCase
Function IHakuswapPair.PERMIT_TYPEHASH() (SwapMining.sol#101) is not in mixedCase
Function IHakuswapPair.MINIMUM_LIQUIDITY() (SwapMining.sol#118) is not in mixedCase
Parameter HakuToken.mintFor(address,uint256)._to (SwapMining.sol#1205) is not in mixedCase
Parameter HakuToken.mintFor(address,uint256)._amount (SwapMining.sol#1205) is not in mixedCase
Constant HakuToken.maxSupply (SwapMining.sol#1203) is not in UPPER_CASE_WITH_UNDERSCORES
Variable HakuToken._delegates (SwapMining.sol#1223) is not in mixedCase
Parameter SwapMining.addPair(uint256,address,bool)._allocPoint (SwapMining.sol#1516) is not in mixedCase
Parameter SwapMining.addPair(uint256,address,bool)._pair (SwapMining.sol#1516) is not in mixedCase
Parameter SwapMining.addPair(uint256,address,bool)._withUpdate (SwapMining.sol#1516) is not in mixedCase
Parameter SwapMining.setPair(uint256,uint256,bool)._pid (SwapMining.sol#1535) is not in mixedCase
Parameter SwapMining.setPair(uint256,uint256,bool)._allocPoint (SwapMining.sol#1535) is not in mixedCase
Parameter SwapMining.setPair(uint256,uint256,bool)._withUpdate (SwapMining.sol#1535) is not in mixedCase
Parameter SwapMining.setHakuswapPerSecond(uint256)._newPerSecond (SwapMining.sol#1544) is not in mixedCase
Parameter SwapMining.addWhitelist(address)._addToken (SwapMining.sol#1550) is not in mixedCase
Parameter SwapMining.delWhitelist(address)._delToken (SwapMining.sol#1555) is not in mixedCase
Parameter SwapMining.isWhitelist(address)._token (SwapMining.sol#1564) is not in mixedCase
Parameter SwapMining.getWhitelist(uint256)._index (SwapMining.sol#1568) is not in mixedCase
Parameter SwapMining.setHalvingPeriod(uint256)._period (SwapMining.sol#1573) is not in mixedCase
Parameter SwapMining.setOracle(IOracle)._oracle (SwapMining.sol#1582) is not in mixedCase
Parameter SwapMining.getHakuReward(uint256)._lastRewardTime (SwapMining.sol#1612) is not in mixedCase
Parameter SwapMining.mint(uint256)._pid (SwapMining.sol#1638) is not in mixedCase
Parameter SwapMining.getUserReward(uint256,address)._pid (SwapMining.sol#1742) is not in mixedCase
Parameter SwapMining.getUserReward(uint256,address)._user (SwapMining.sol#1742) is not in mixedCase
Parameter SwapMining.getPoolInfo(uint256)._pid (SwapMining.sol#1757) is not in mixedCase
Parameter SwapMining.ownerWithdraw(address,uint256)._to (SwapMining.sol#1770) is not in mixedCase
Parameter SwapMining.ownerWithdraw(address,uint256)._amount (SwapMining.sol#1770) is not in mixedCase
Parameter SwapMining.safeHakuTransfer(address,uint256)._to (SwapMining.sol#1774) is not in mixedCase
Parameter SwapMining.safeHakuTransfer(address,uint256)._amount (SwapMining.sol#1774) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (SwapMining.sol#858)" inContext (SwapMining.sol#852-861)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (SwapMining.sol#899-902)
transferOwnership(address) should be declared external:
```

```
decimals() should be declared external:
        - ERC20.decimals() (SwapMining.sol#961-963)
symbol() should be declared external:
        - ERC20.symbol() (SwapMining.sol#968-970)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (SwapMining.sol#994-997)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (SwapMining.sol#1002-1004)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (SwapMining.sol#1013-1016)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (SwapMining.sol#1030-1042)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (SwapMining.sol#1056-1059)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (SwapMining.sol#1075-1082)
mint(uint256) should be declared external:
        - ERC20.mint(uint256) (SwapMining.sol#1092-1095)
        - HakuToken.mint(uint256) (SwapMining.sol#1211-1215)
mintFor(address,uint256) should be declared external:
        - HakuToken.mintFor(address,uint256) (SwapMining.sol#1205-1209)
addPair(uint256,address,bool) should be declared external:
        - SwapMining.addPair(uint256,address,bool) (SwapMining.sol#1516-1532)
setPair(uint256,uint256,bool) should be declared external:
        - SwapMining.setPair(uint256,uint256,bool) (SwapMining.sol#1535-1541)
setHakuswapPerSecond(uint256) should be declared external:
        - SwapMining.setHakuswapPerSecond(uint256) (SwapMining.sol#1544-1547)
addWhitelist(address) should be declared external:
        - SwapMining.addWhitelist(address) (SwapMining.sol#1550-1553)
delWhitelist(address) should be declared external:
        - SwapMining.delWhitelist(address) (SwapMining.sol#1555-1558)
setHalvingPeriod(uint256) should be declared external:
        - SwapMining.setHalvingPeriod(uint256) (SwapMining.sol#1573-1575)
setRouter(address) should be declared external:
        - SwapMining.setRouter(address) (SwapMining.sol#1577-1580)
setOracle(IOracle) should be declared external:
        - SwapMining.setOracle(IOracle) (SwapMining.sol#1582-1585)
phase() should be declared external:
```

```
phase() should be declared external:
        - SwapMining.phase() (SwapMining.sol#1598-1600)
reward() should be declared external:
        - SwapMining.reward() (SwapMining.sol#1607-1609)
swap(address,address,address,uint256) should be declared external:
        - SwapMining.swap(address,address,address,uint256) (SwapMining.sol#1661-1694)
takerWithdraw() should be declared external:
        - SwapMining.takerWithdraw() (SwapMining.sol#1717-1739)
getUserReward(uint256,address) should be declared external:
        - SwapMining.getUserReward(uint256,address) (SwapMining.sol#1742-1754)
getPoolInfo(uint256) should be declared external:
        - SwapMining.getPoolInfo(uint256) (SwapMining.sol#1757-1768)
ownerWithdraw(address,uint256) should be declared external:
        - SwapMining.ownerWithdraw(address,uint256) (SwapMining.sol#1770-1772)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SwapMining.sol analyzed (13 contracts with 75 detectors), 115 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```


Copyright © 2021
Hash0X
(audit@hash0x.com)

Slither log >> MockToken.sol

```
INFO:Detectors:
ERC20.allowance(address,address).owner (MockToken.sol#665) shadows:
        - Ownable.owner() (MockToken.sol#543-545) (function)
ERC20._approve(address,address,uint256).owner (MockToken.sol#837) shadows:
        - Ownable.owner() (MockToken.sol#543-545) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Address.isContract(address) (MockToken.sol#292-303) uses assembly
        - INLINE ASM (MockToken.sol#299-301)
Address._functionCallWithValue(address,bytes,uint256,string) (MockToken.sol#400-426) uses assembly
        - INLINE ASM (MockToken.sol#418-421)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (MockToken.sol#400-426) is never used and should be removed
Address.functionCall(address,bytes) (MockToken.sol#347-349) is never used and should be removed
Address.functionCall(address,bytes,string) (MockToken.sol#357-363) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (MockToken.sol#376-382) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (MockToken.sol#390-398) is never used and should be removed
Address.isContract(address) (MockToken.sol#292-303) is never used and should be removed
Address.sendValue(address,uint256) (MockToken.sol#321-327) is never used and should be removed
Context._msgData() (MockToken.sol#520-523) is never used and should be removed
ERC20._burn(address,uint256) (MockToken.sol#815-821) is never used and should be removed
ERC20._burnFrom(address,uint256) (MockToken.sol#854-861) is never used and should be removed
SafeERC20._callOptionalReturn(IERC20,bytes) (MockToken.sol#500-511) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (MockToken.sol#457-471) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (MockToken.sol#482-492) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (MockToken.sol#473-480) is never used and should be removed
SafeERC20.safeTransfer(IERC20,address,uint256) (MockToken.sol#433-439) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (MockToken.sol#441-448) is never used and should be removed
SafeMath.div(uint256,uint256) (MockToken.sol#93-95) is never used and should be removed
SafeMath.div(uint256,uint256,string) (MockToken.sol#109-119) is never used and should be removed
SafeMath.min(uint256,uint256) (MockToken.sol#158-160) is never used and should be removed
SafeMath.mod(uint256,uint256) (MockToken.sol#133-135) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (MockToken.sol#149-156) is never used and should be removed
SafeMath.mul(uint256,uint256) (MockToken.sol#67-79) is never used and should be removed
SafeMath.sqrt(uint256) (MockToken.sol#163-174) is never used and should be removed
SafeMath.sub(uint256,uint256) (MockToken.sol#32-34) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
INFO:Detectors:
Pragma version>=0.6.6 (MockToken.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (MockToken.sol#321-327):
        - (success) = recipient.call{value: amount}() (MockToken.sol#325)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (MockToken.sol#400-426):
        - (success,returndata) = target.call{value: weiValue}(data) (MockToken.sol#409)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Redundant expression "this (MockToken.sol#521)" inContext (MockToken.sol#515-524)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (MockToken.sol#562-565)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (MockToken.sol#571-575)
name() should be declared external:
        - ERC20.name() (MockToken.sol#617-619)
decimals() should be declared external:
        - ERC20.decimals() (MockToken.sol#624-626)
symbol() should be declared external:
        - ERC20.symbol() (MockToken.sol#631-633)
totalSupply() should be declared external:
        - ERC20.totalSupply() (MockToken.sol#638-640)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (MockToken.sol#645-647)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (MockToken.sol#657-660)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (MockToken.sol#665-667)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (MockToken.sol#676-679)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (MockToken.sol#693-705)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (MockToken.sol#719-722)
decreaseAllowance(address,uint256) should be declared external:
```

```
mint(uint256) should be declared external:
        - ERC20.mint(uint256) (MockToken.sol#755-758)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MockToken.sol analyzed (8 contracts with 75 detectors), 46 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```


Copyright © 2021
Hash0X
(audit@hash0x.com)

Slither log >> Pair.sol

```
INFO:Detectors:
HakuswapPair.initialize(address,address)._token0 (Pair.sol#480) lacks a zero-check on :
                - token0 = _token0 (Pair.sol#482)
HakuswapPair.initialize(address,address)._token1 (Pair.sol#480) lacks a zero-check on :
                - token1 = _token1 (Pair.sol#483)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in HakuswapPair.burn(address) (Pair.sol#548-570):
        External calls:
        - _safeTransfer(_token0,to,amount0) (Pair.sol#562)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Pair.sol#459)
        - _safeTransfer(_token1,to,amount1) (Pair.sol#563)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Pair.sol#459)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (Pair.sol#567)
                - price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (Pair.sol#493)
        - _update(balance0,balance1,_reserve0,_reserve1) (Pair.sol#567)
                - price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (Pair.sol#494)
Reentrancy in HakuswapPair.swap(uint256,uint256,address,bytes) (Pair.sol#573-601):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (Pair.sol#584)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Pair.sol#459)
        - _safeTransfer(_token1,to,amount1Out) (Pair.sol#585)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Pair.sol#459)
        - IHakuswapCallee(to).hakuswapCall(msg.sender,amount0Out,amount1Out,data) (Pair.sol#586)
        State variables written after the call(s):
        - _update(balance0,balance1,_reserve0,_reserve1) (Pair.sol#599)
                - price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv(_reserve0)) * timeElapsed (Pair.sol#493)
        - _update(balance0,balance1,_reserve0,_reserve1) (Pair.sol#599)
                - price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv(_reserve1)) * timeElapsed (Pair.sol#494)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in HakuswapPair.burn(address) (Pair.sol#548-570):
        External calls:
        - _safeTransfer(_token0,to,amount0) (Pair.sol#562)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Pair.sol#459)
        - _safeTransfer(_token1,to,amount1) (Pair.sol#563)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Pair.sol#459)
```

```
        Event emitted after the call(s):
        - Burn(msg.sender,amount0,amount1,to) (Pair.sol#569)
        - Sync(reserve0,reserve1) (Pair.sol#499)
                - _update(balance0,balance1,_reserve0,_reserve1) (Pair.sol#567)
Reentrancy in HakuswapPair.swap(uint256,uint256,address,bytes) (Pair.sol#573-601):
        External calls:
        - _safeTransfer(_token0,to,amount0Out) (Pair.sol#584)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Pair.sol#459)
        - _safeTransfer(_token1,to,amount1Out) (Pair.sol#585)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Pair.sol#459)
        - IHakuswapCallee(to).hakuswapCall(msg.sender,amount0Out,amount1Out,data) (Pair.sol#586)
        Event emitted after the call(s):
        - Swap(msg.sender,amount0In,amount1In,amount0Out,amount1Out,to) (Pair.sol#600)
        - Sync(reserve0,reserve1) (Pair.sol#499)
                - _update(balance0,balance1,_reserve0,_reserve1) (Pair.sol#599)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
HakuswapERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (Pair.sol#375-387) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,Hakuswap: EXPIRED) (Pair.sol#376)
HakuswapPair._update(uint256,uint256,uint112,uint112) (Pair.sol#487-500) uses timestamp for comparisons
        Dangerous comparisons:
        - timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0 (Pair.sol#491)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
HakuswapERC20.constructor() (Pair.sol#318-332) uses assembly
        - INLINE ASM (Pair.sol#320-322)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Context._msgData() (Pair.sol#203-206) is never used and should be removed
SafeMath.div(uint256,uint256) (Pair.sol#94-96) is never used and should be removed
SafeMath.div(uint256,uint256,string) (Pair.sol#110-120) is never used and should be removed
SafeMath.min(uint256,uint256) (Pair.sol#159-161) is never used and should be removed
SafeMath.mod(uint256,uint256) (Pair.sol#134-136) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (Pair.sol#150-157) is never used and should be removed
SafeMath.sqrt(uint256) (Pair.sol#164-175) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
```

```
INFO:Detectors:
Pragma version>=0.6.6 (Pair.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in HakuswapPair._safeTransfer(address,address,uint256) (Pair.sol#458-461):
        - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (Pair.sol#459)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IHakuswapERC20.DOMAIN_SEPARATOR() (Pair.sol#293) is not in mixedCase
Function IHakuswapERC20.PERMIT_TYPEHASH() (Pair.sol#294) is not in mixedCase
Variable HakuswapERC20.DOMAIN_SEPARATOR (Pair.sol#310) is not in mixedCase
Parameter HakuswapPair.initialize(address,address)._token0 (Pair.sol#480) is not in mixedCase
Parameter HakuswapPair.initialize(address,address)._token1 (Pair.sol#480) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (Pair.sol#204)" inContext (Pair.sol#198-207)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable HakuswapPair.swap(uint256,uint256,address,bytes).balance0Adjusted (Pair.sol#594) is too similar to HakuswapPair.swap(uint256,uin
t256,address,bytes).balance1Adjusted (Pair.sol#595)
Variable HakuswapPair.price0CumulativeLast (Pair.sol#440) is too similar to HakuswapPair.price1CumulativeLast (Pair.sol#441)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Pair.sol#245-248)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Pair.sol#254-258)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Pair.sol analyzed (11 contracts with 75 detectors), 33 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```


Copyright © 2021
Hash0X
(audit@hash0x.com)

Hash0X

Copyright © 2021
Hash0X
(audit@hash0x.com)