Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: Voy Finance
Website: http://Voyfinance.com
Platform: Ethereum
Language: Solidity
Date: November 2nd, 2022

Table of contents

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

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by Voy Finance to perform the Security audit of the Voy Finance protocol smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on November 2nd, 2022.

The purpose of this audit was to address the following:

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

Project Background

Voy Finance Contract is a Pooling DeFi to trade digitalisation and ESG smart contract, having functions like mint, burn, stake, unStake, deposit, migrate, pause, unpause, claim, withdraw, withdrawAll, etc. The Voy Finance contract inherits Ownable, ERC20, IERC20, SafeERC20, Pausable, SafeMath, AccessControl standard smart contracts from the OpenZeppelin library. These OpenZeppelin contracts are considered community audited and time tested, and hence are not part of the audit scope.

Audit scope

Name	Code Review and Security Analysis Report for Voy Finance Protocol Smart Contracts
Platform	**Ethereum / Solidity**
File 1	VoySale.sol
File 1 MD5 Hash	DAD18FE9557684BA5330D4C31E11F1BE
Updated File 1 MD5 Hash	0F18C4485FEFB59993230C7AEED70A76
File 2	VoyToken.sol
File 2 MD5 Hash	11E4EA4CFF80DE5028C0E55C6855DCA3
Updated File 2 MD5 Hash	EF9A76785663D0B31B022AD913A5C9C2
File 3	VoyXDCToken.sol
File 3 MD5 Hash	768462FE08C85E935FBD4F5FF26254DE
Updated File 3 MD5 Hash	2CBC8F4EB6382F40B2A2FD610D2760F4
File 4	XDCBridge.sol
File 4 MD5 Hash	EC4C73E3A928A17161D92CFF5A1CDA65
File 5	MasterChef.sol
File 5 MD5 Hash	3D643CC5B29A43C837B15B2164F1A1F4
Updated File 5 MD5 Hash	0083376AAFDFA2BECB7C50E78B513FA5
File 6	VoyStaking.sol
File 6 MD5 Hash	3263033B48E19BCFBC7A757AAD318D43
Updated File 6 MD5 Hash	496CC77BEA00ADEE840414A713700D86
File 7	VoyVesting.sol
File 7 MD5 Hash	05E8EE06EED46194AC419D1A9C6FF7D5
Audit Date	November 2nd, 2022
Revise Audit Date	November 4th, 2022

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 VoySale.sol • Owner can set Whitelist addresses. • Owner can recover tokens and ether.	**YES, This is valid.**
File 2 XDCBridge.sol • Owner can return the swap. • Owner can initiate a swap.	**YES, This is valid.**
File 3 VoyXDCToken.sol • Name: Voy Token • Symbol: VOY • Owner can mint unlimited tokens.	**YES, This is valid.**
File 4 MasterChef.sol • Owner can add a new LP to the pool. • Owner can update the given pool's VOY allocation point. • Owner can set the migrator contract.	**YES, This is valid.**
File 5 VoyStaking.sol • Owner can set a Harvest Fee. • Owner can set an UnStake Fee. • Owner can add and remove an UnStake Fee.	**YES, This is valid.**
File 6 VoyToken.sol • Name: Voy Token • Symbol: VOY • Initial supply: 40 Million • Total Max Supply: 500 Million • The VoyToken contract implements a continuous minting function.	**YES, This is valid.**
File 7 VoyVesting.sol • Vesting Period: 180 Days	**YES, This is valid.**

● Owner can set the Vesting Period. ● Owner can withdraw all tokens from the account. ● Users can claim the amount.	

Audit Summary

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

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

We found 0 critical, 2 high, 1 medium and 8 low and some very low level issues.
All the issues have been fixed / acknowledged in the revised code.

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

Technical Quick Stats

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

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 7 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

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

The Voy Finance team has provided unit test scripts, which would have helped to determine the integrity of the code in an automated way.

All code parts are not well commented on smart contracts.

Documentation

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

As mentioned above, code parts are not well commented. But the logic is straightforward. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its website http://Voyfinance.com which provided rich information about the project architecture.

Use of Dependencies

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

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

AS-IS overview

VoySale.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	setWhitelist	write	access only Owner	No Issue
7	recoverTokens	write	access only Owner	No Issue
8	recoverETH	write	access only Owner	No Issue
9	buy	write	Hardcoded token addresses	Refer Audit Findings

VoyToken.sol

Functions

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

26	grantRole	write	access only Role	No Issue
27	revokeRole	write	access only Role	No Issue
28	renounceRole	write	Passed	No Issue
29	_setupRole	internal	Passed	No Issue
30	_setRoleAdmin	internal	Passed	No Issue
31	_grantRole	internal	Passed	No Issue
32	_revokeRole	internal	Passed	No Issue
33	mint	external	access only Role	No Issue
34	burn	external	access only Role	No Issue
35	setBridge	external	access only Role	No Issue
36	isContract	internal	Passed	No Issue
37	swap	external	Passed	No Issue
38	_beforeTokenTransfer	internal	Passed	No Issue

XDCBridge.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	whenNotPaused	modifier	Passed	No Issue
7	whenPaused	modifier	Passed	No Issue
8	paused	read	Passed	No Issue
9	_requireNotPaused	internal	Passed	No Issue
10	_requirePaused	internal	Passed	No Issue
11	_pause	internal	Passed	No Issue
12	_unpause	internal	Passed	No Issue
13	initiateSwap	external	Passed	No Issue
14	returnSwap	external	access only Owner	No Issue
15	pause	external	access only Owner	No Issue
16	unpause	external	access only Owner	No Issue

VoyXDCToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	owner	read	Passed	No Issue
2	onlyOwner	modifier	Passed	No Issue
3	renounceOwnership	write	access only Owner	No Issue
4	transferOwnership	write	access only Owner	No Issue
5	getOwner	external	Passed	No Issue
6	name	read	Passed	No Issue

7	decimals	read	Passed	No Issue
8	symbol	read	Passed	No Issue
9	totalSupply	read	Passed	No Issue
10	balanceOf	read	Passed	No Issue
11	transfer	write	Passed	No Issue
12	allowance	read	Passed	No Issue
13	approve	write	Passed	No Issue
14	transferFrom	write	Passed	No Issue
15	increaseAllowance	write	Passed	No Issue
16	decreaseAllowance	write	Passed	No Issue
17	mint	write	Unlimited minting	Refer Audit Findings
18	_transfer	internal	Passed	No Issue
19	_mint	internal	Passed	No Issue
20	_burn	internal	Passed	No Issue
21	_approve	internal	Passed	No Issue
22	_burnFrom	internal	Passed	No Issue
23	completeSwapFromEthereum	external	access only Owner	No Issue
24	swapToEthereum	external	Passed	No Issue

MasterChef.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	poolLength	external	Passed	No Issue
7	add	write	access only Owner	No Issue
8	set	write	Passed	No Issue
9	setMigrator	write	access only Owner	No Issue
10	migrate	write	Critical operation lacks event log	Refer Audit Findings
11	getMultiplier	read	Passed	No Issue
12	pendingVoy	external	Passed	No Issue
13	massUpdatePools	write	Passed	No Issue
14	updatePool	write	Passed	No Issue
15	deposit	write	Passed	No Issue
16	withdraw	write	Passed	No Issue
17	emergencyWithdraw	write	Passed	No Issue
18	safeVoyTransfer	internal	Passed	No Issue

VoyStaking.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	init	write	Passed	No Issue
7	setFeeWallet	external	access only Owner	No Issue
8	setUnStakeFee	external	access only Owner	No Issue
9	addUnStakeFee	external	Empty unStakeFees array issue	Refer Audit Findings
10	removeUnStakeFee	external	access only Owner	No Issue
11	setHarvestFee	external	access only Owner	No Issue
12	depositReward	external	access only Owner	No Issue
13	stake	external	Passed	No Issue
14	unStake	external	Passed	No Issue
15	updateUserStatus	write	Passed	No Issue
16	getMaxUnStakeFeeDays	write	Passed	No Issue
17	harvest	external	Passed	No Issue
18	getMultiplier	external	Passed	No Issue
19	_getMultiplier	internal	Passed	No Issue
20	getPending	external	Passed	No Issue
21	_getPending	read	Passed	No Issue
22	getRewardBalance	external	Passed	No Issue
23	_updateStatus	write	Passed	No Issue
24	_getUnStakeFeePercent	internal	Passed	No Issue

VoyVesting.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	whenStartedVestingSeason	modifier	Passed	No Issue
7	whenNotStartedVestingSeason	modifier	Passed	No Issue
8	whenFinishedVestingPeriod	modifier	Passed	No Issue
9	onlyUser	modifier	Passed	No Issue
10	setVestingPeriod	external	access only Owner	No Issue
11	deposit	external	access only Owner	No Issue
12	addUsers	external	access only Owner	No Issue
13	addUser	external	access only Owner	No Issue

14	_addUser	write	access only Owner	No Issue
15	stake	external	access only Owner	No Issue
16	withdraw	external	access only Owner	No Issue
17	withdrawAll	external	access only Owner	No Issue
18	claim	external	Passed	No Issue
19	getClaimAmount	external	Passed	No Issue
20	_getClaimAmount	internal	Passed	No Issue

Severity Definitions

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

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

(1) Claim function not working: **VoyVesting.sol**

```
// ============================== User Functions ==============================
function claim() external whenFinishedVestingPeriod onlyUser {   infinite gas
    User storage user = users[msg.sender];
    require(user.claimAmount == 0, "Already claim");
    uint256 lockedAmount = user.lockedAmount;
    uint256 unstakedAmount = stakingContract.unStake(lockedAmount);

    uint256 rewardAmount = stakingContract.harvest(false);
    totalRewardAmount += rewardAmount;
    uint256 userRewardAmount = totalRewardAmount * lockedAmount / totalVestedAmount;
```

The voyvesting claim function is not working. The error occurs while the voystaking contract has a harvest function which doesn't have any parameters.

Resolution: We advise to re-check the code and use appropriate staking contracts to claim.

Status: Fixed

(2) Need approval to transfer the tokens: **VoySale.sol**

```
function recoverTokens(uint256 _amount, address _token) public onlyOwner {
    ERC20(_token).transferFrom(address(this), msg.sender, _amount);
}
```

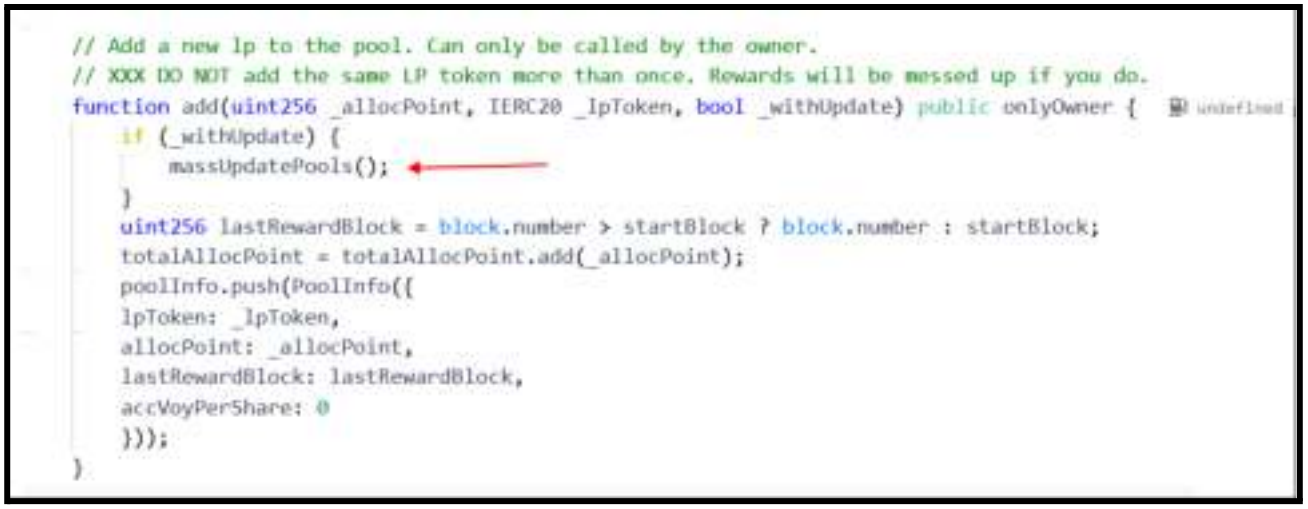
In the recoverTokens function, transferFrom is used to transfer tokens from contract to the owner, which needs approval of the contract to the owner. But no approval has been given, so the token transfer is not working.

Resolution: We suggest using the "transfer" function instead of "transferFrom" function Or add code to give approval before the "transferFrom" function.

Status: Fixed

Medium

(1) Duplicate LP token: **Masterchef.sol**



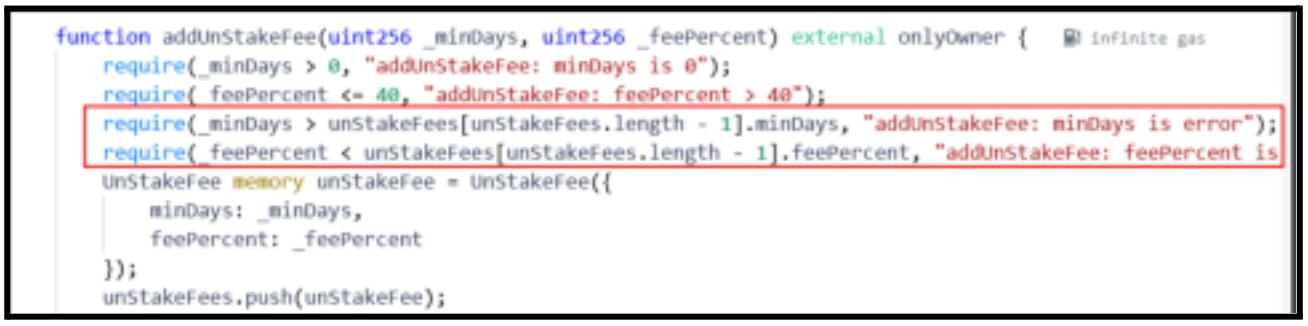
```
// Add a new lp to the pool. Can only be called by the owner.
// XXX DO NOT add the same LP token more than once. Rewards will be messed up if you do.
function add(uint256 _allocPoint, IERC20 _lpToken, bool _withUpdate) public onlyOwner {
    if (_withUpdate) {
        massUpdatePools();
    }
    uint256 lastRewardBlock = block.number > startBlock ? block.number : startBlock;
    totalAllocPoint = totalAllocPoint.add(_allocPoint);
    poolInfo.push(PoolInfo({
    lpToken: _lpToken,
    allocPoint: _allocPoint,
    lastRewardBlock: lastRewardBlock,
    accVoyPerShare: 0
    }));
}
```

In add function, users can add the same lp token multiple times. The rewards will be messed up if you add a duplicate LP token.

Resolution: We advise to put validation to check for duplicate LP tokens.
Status: Fixed

Low

(1) Empty unStakeFees array issue: **VoyStaking.sol**



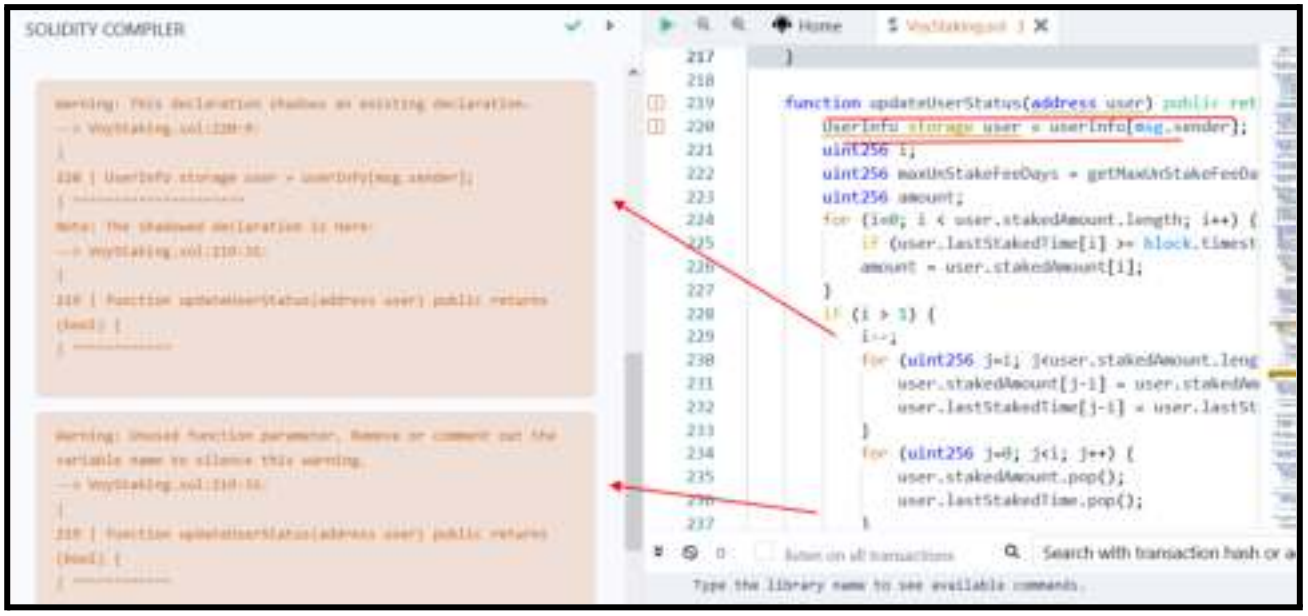
```
function addUnStakeFee(uint256 _minDays, uint256 _feePercent) external onlyOwner {
    require(_minDays > 0, "addUnStakeFee: minDays is 0");
    require(_feePercent <= 40, "addUnStakeFee: feePercent > 40");
    require(_minDays > unStakeFees[unStakeFees.length - 1].minDays, "addUnStakeFee: minDays is error");
    require(_feePercent < unStakeFees[unStakeFees.length - 1].feePercent, "addUnStakeFee: feePercent is
    UnStakeFee memory unStakeFee = UnStakeFee({
        minDays: _minDays,
        feePercent: _feePercent
    });
    unStakeFees.push(unStakeFee);
```

In addUnStakeFee function, there are 2 required conditions which uses unStakeFees.length - 1 to validate mindays and feePercentage. Here, if the unStakeFees

array is empty then unStakeFees.length -1 will throw an error and execution will be reverted.

Resolution: We suggest putting conditions before these 2 require statements to check unStakeFees.length should be greater than 0.
Status: Acknowledged

(2) Other Programming Issue: **VoyStaking.sol**



In updateUserStatus function, function parameter and defined variable both have the same name.This declaration shadows an existing declaration and Unused function parameter.

Resolution: We suggest to either change the variable name for UserInfo Or function parameter name.
Status: Fixed

(3) Function input parameters lack of check:

Variable validation is not performed in below functions:
VoyToken.sol

- swap = _bridge

Resolution: We advise to put validation: integer type variables should be greater than 0 and address type variables should not be address(0).

Status: Fixed

(4) Other Programming Issue: **VoyStaking.sol**

In getMaxUnStakeFeeDays function, Function state mutability can be restricted to view.

Resolution: We suggest adding the "view" keyword after the "public" keyword.

Status: Fixed

(5) Unlimited minting: **VoyXDCToken.sol**

Owner can mint unlimited tokens.

Resolution: We suggest using a minting limit.

Status: Acknowledged

(6) Critical operation lacks event log: **Masterchef.sol**

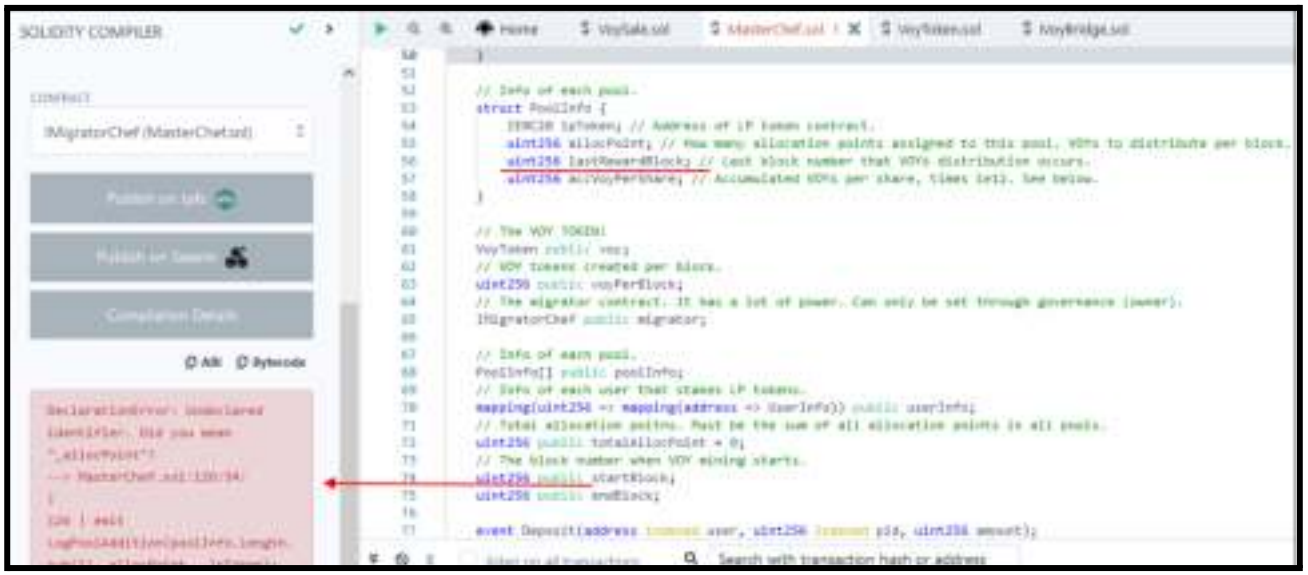
Missing event log for:

- add
- set
- migrate
- updatePool

Resolution: Write an event log for listed events.

Status: Fixed

(7) Event parameter mismatched: **Masterchef.sol**

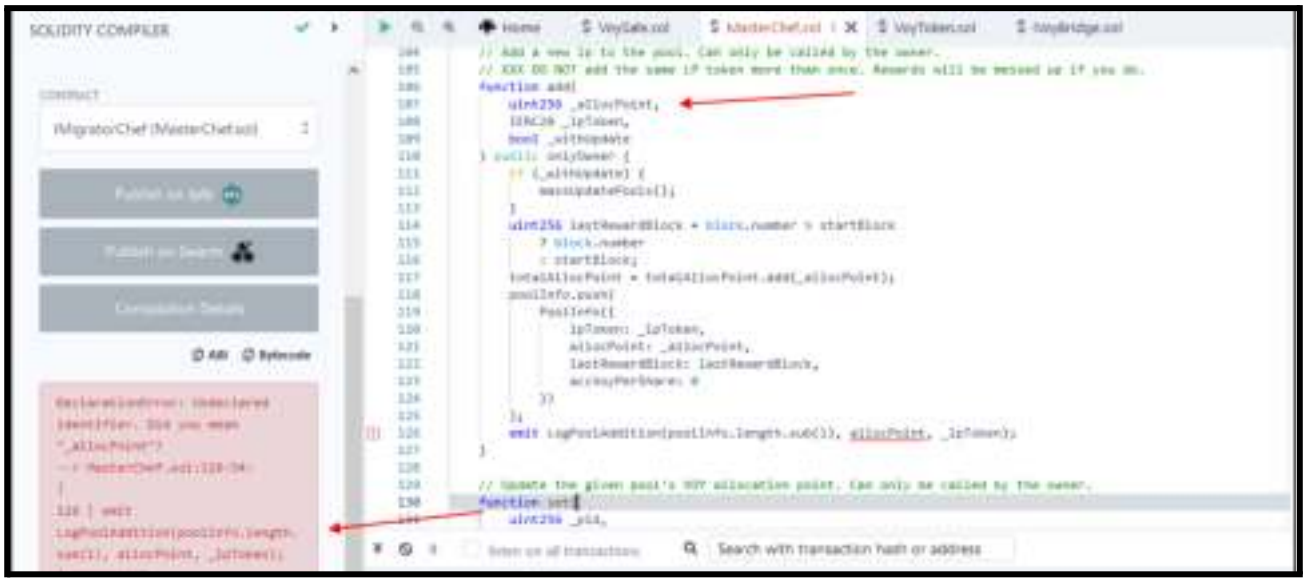


In LogUpdatePool event, lastRewardBlock parameter has a uint64 type which does not match with pool info element lastRewardBlock.

Resolution: We suggest changing the lastRewardBlock parameter type in the LogUpdatePool event.

Status: Fixed

(8) Wrong parameter name: **Masterchef.sol**

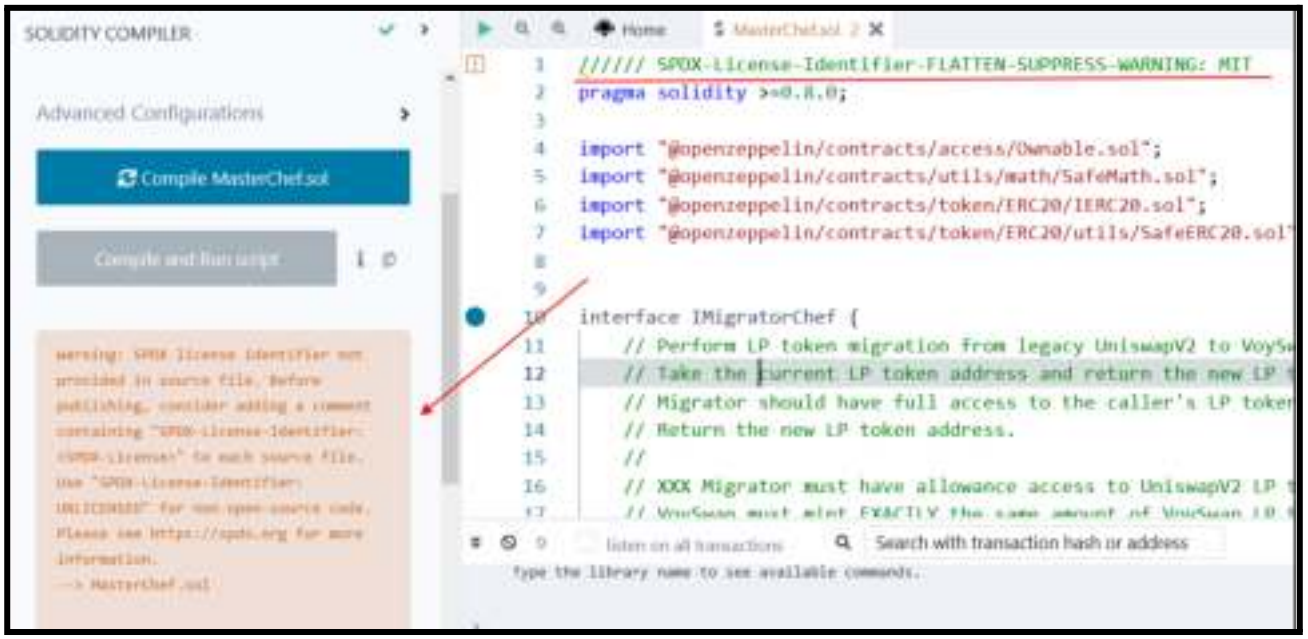


In add function, LogPoolAddition emit has wrong parameter "allocPoint".

Resolution: We suggest changing the parameter name from "allocPoint" to "_allocPoint".
Status: Fixed

Very Low / Informational / Best practices:

(1) Missing SPDX-License-Identifier: **Masterchef.sol**



SPDX-License-Identifier is written but with wrong syntax.

Resolution: We advise adding the correct SPDX License Identifier.
Status: **Fixed**

(2) Hardcoded token addresses: **VoySale.sol**

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

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

(3) Consider specifying function visibility to “external” instead of “public”, if that function is not being called internally. It will save some gas as well.
https://ethereum.stackexchange.com/questions/32353/what-is-the-difference-between-an-internal-external-and-public-private-function/32464
Status: Acknowledged

Centralization

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

- add: MasterChef owner can add a new lp to the pool.
- set: MasterChef owner can update the given pool's VOY allocation point.
- setMigrator: MasterChef owner can set the migrator contract.
- setFeeWallet: VoyStaking owner can set fee wallet address.
- setUnStakeFee: VoyStaking owner can set an unstake fee percentage.
- addUnStakeFee: VoyStaking owner can add an unstake fee percentage.
- removeUnStakeFee: VoyStaking owner can remove unstake fee percentage.
- setHarvestFee: VoyStaking owner can set harvest fee percentage.
- depositReward: VoyStaking owner can deposit reward amount.
- stake: VoyStaking owner can stake amount.
- setVestingPeriod: VoyVesting owner can set vesting period.
- deposit: VoyVesting owner can deposit amount.
- addUsers: VoyVesting owner can add new users addresses.
- addUser: VoyVesting owner can add a new user address.
- _addUser: VoyVesting owner can add a new user address.
- stake: VoyVesting owner can stake amount.
- withdraw: VoyVesting owners can withdraw amounts.
- withdrawAll: VoyVesting owners can withdraw all maximum amounts from account.
- claim: VoyVesting owners can claim the amount.
- setWhitelist: VoySale owners can set whitelist addresses.
- recoverTokens: VoySale owners can recover tokens from addresses.
- recoverETH: VoySale owners can recover ether from addresses.
- mint: VoyToken role owners can mint tokens to addresses.
- burn: VoyToken role owners can burn tokens from addresses.
- setBridge: VoyToken role owners can set bridge address enabled status true.
- completeSwapFromEthereum: VoyXDCToken owner can complete swap from ethereum address.
- initiateSwap: XDCBridge owner can initiate swap address.

- returnSwap: XDCBridge owner can return swap address.
- pause: XDCBridge owner can trigger stopped state.
- unpause: XDCBridge owner can return to normal state.

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

Conclusion

We were given a contract code in the form of a file. And we have used all possible tests based on given objects as files. We have observed 2 high severity issues, 1 medium severity issue, 8 low severity issues and some informational issues in the smart contracts. All the issues have been fixed / acknowledged in the revised code. So, **it's good to go for the mainnet deployment**.

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

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

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

Security state of the reviewed contract, based on standard audit procedure scope, is **"Secure".**

Our Methodology

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

Manual Code Review:

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

Vulnerability Analysis:

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

Documenting Results:

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

Suggested Solutions:

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

Disclaimers

EtherAuthority.io Disclaimer

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

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

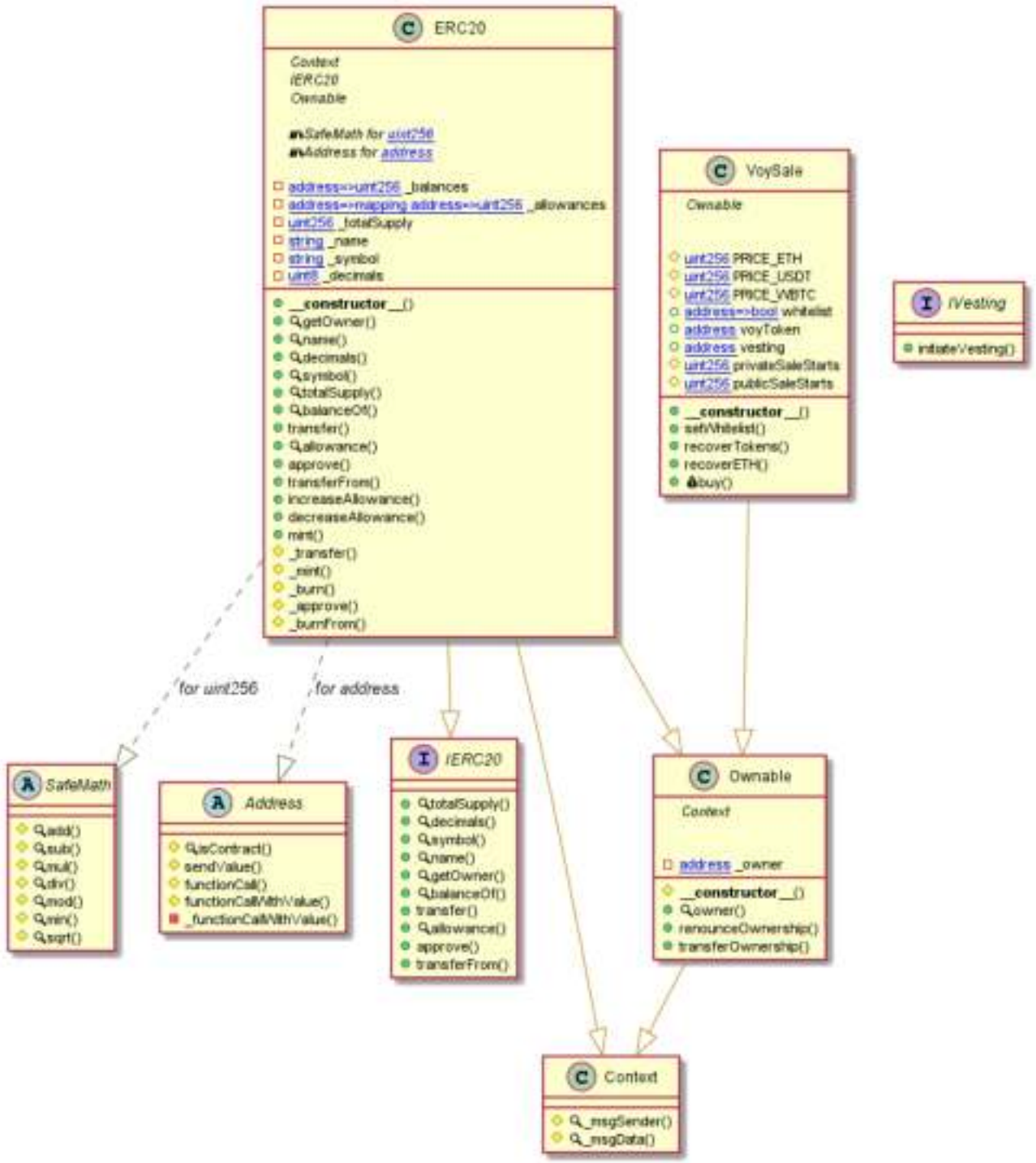
Technical Disclaimer

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

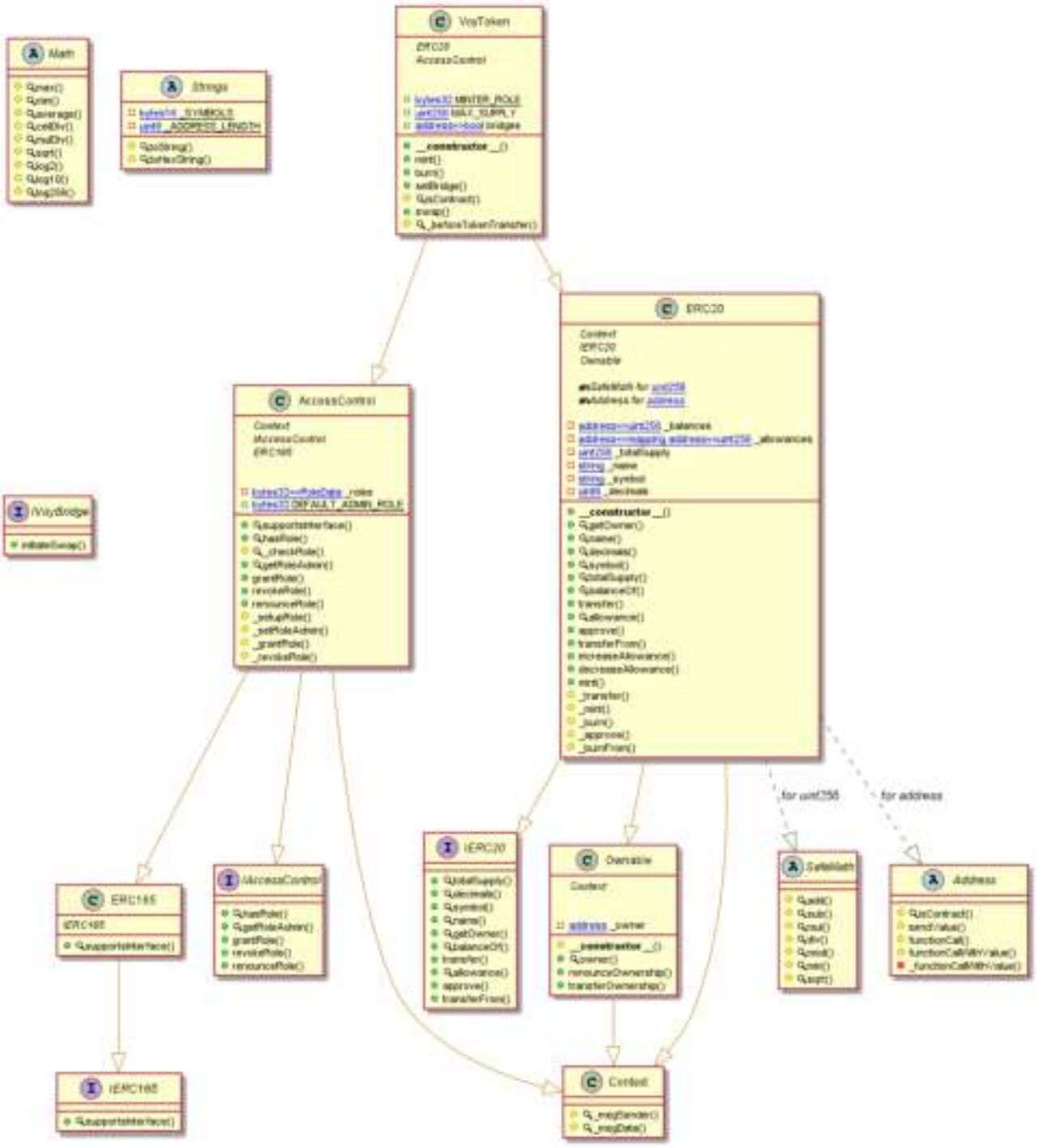
Appendix

Code Flow Diagram - Voy Finance Protocol

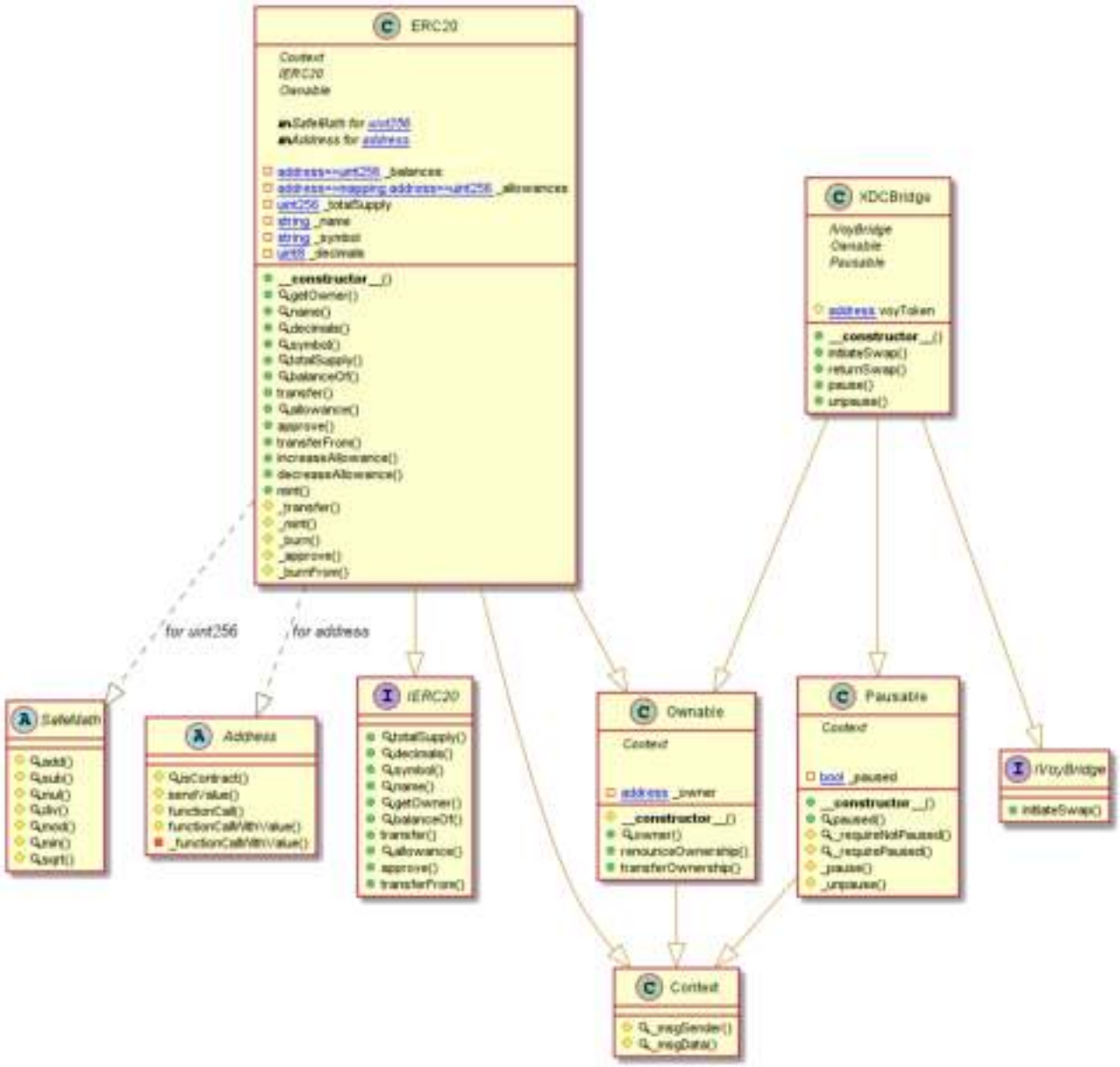
VoySale Diagram



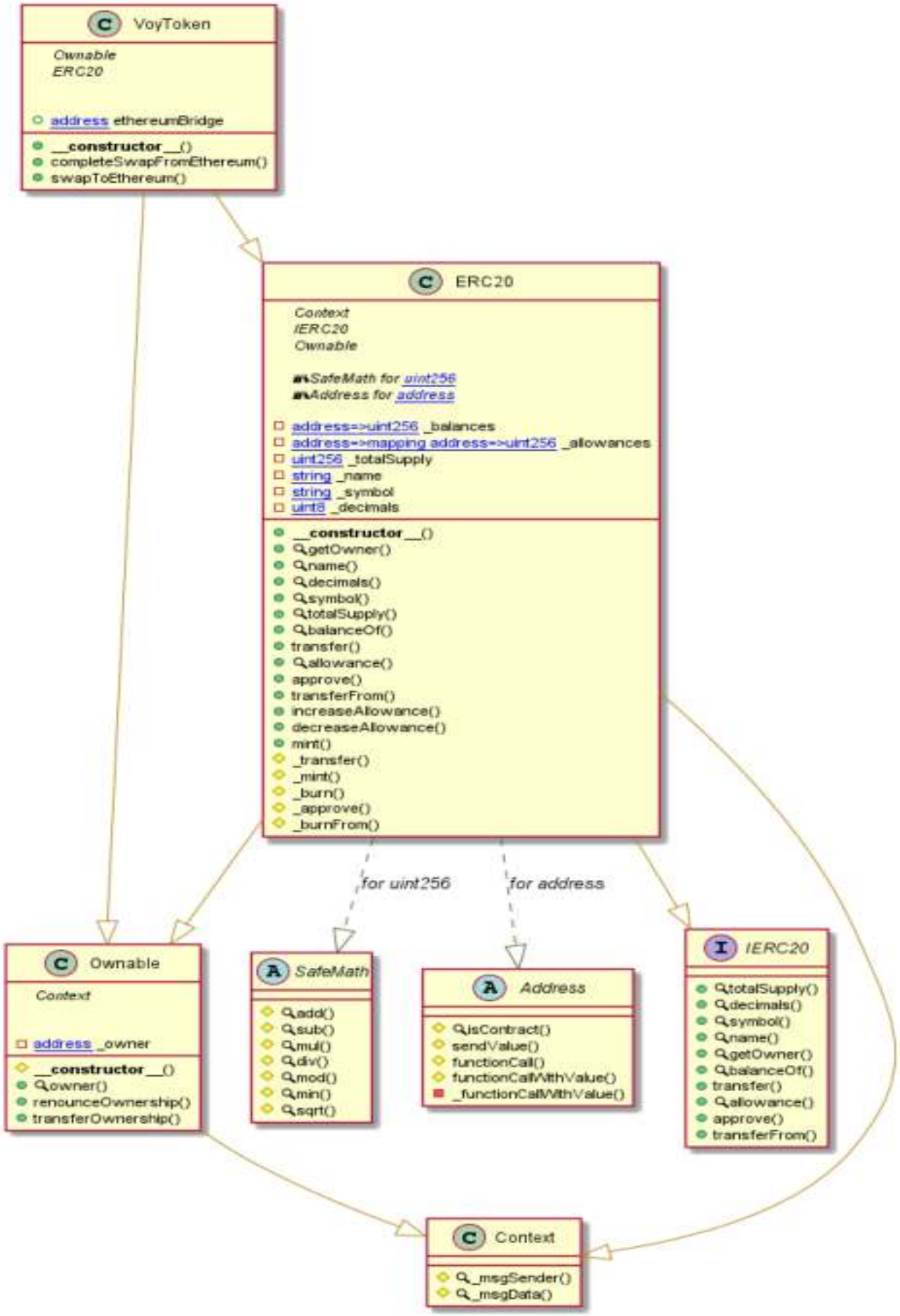
VoyToken Diagram



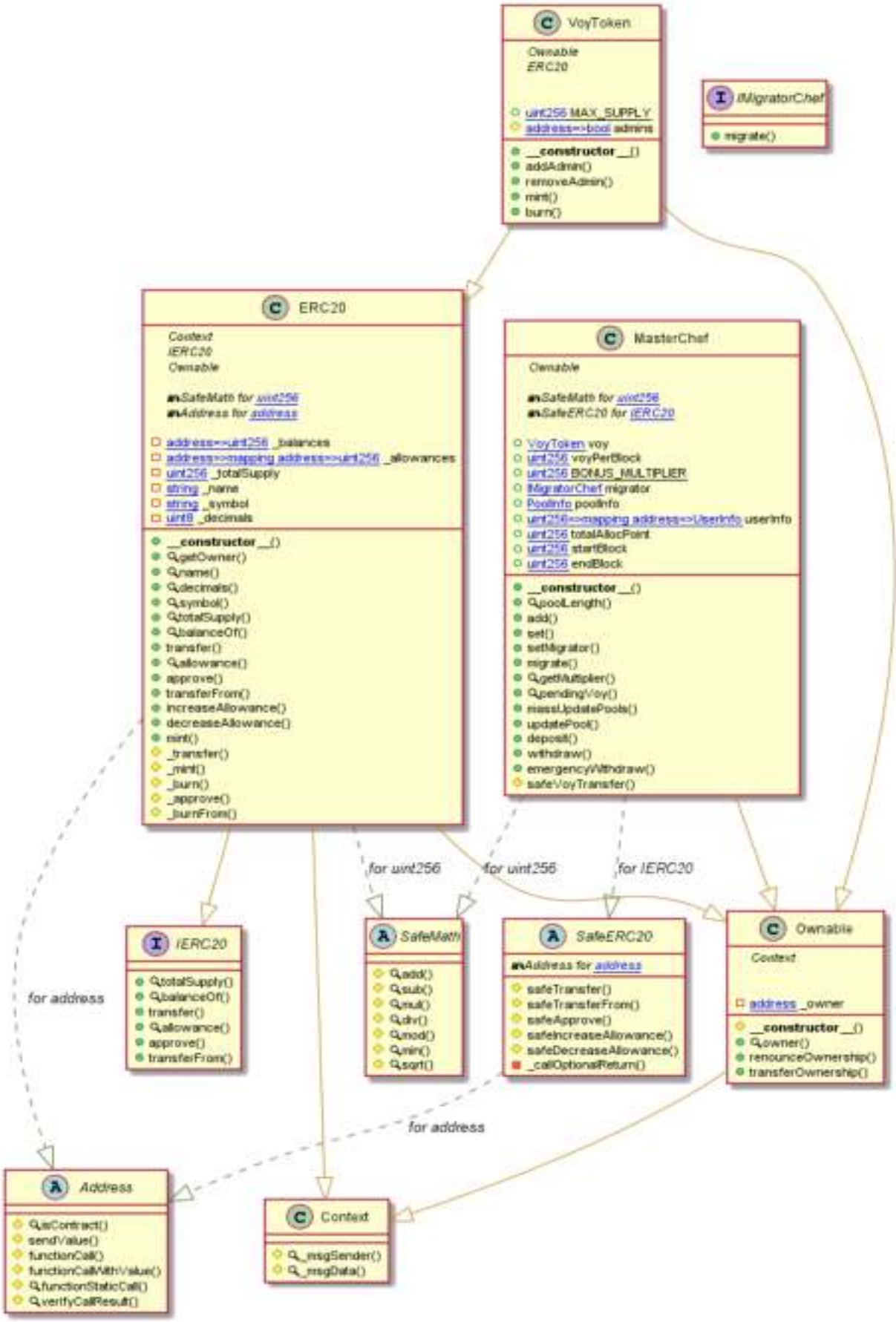
XDCBridge Diagram



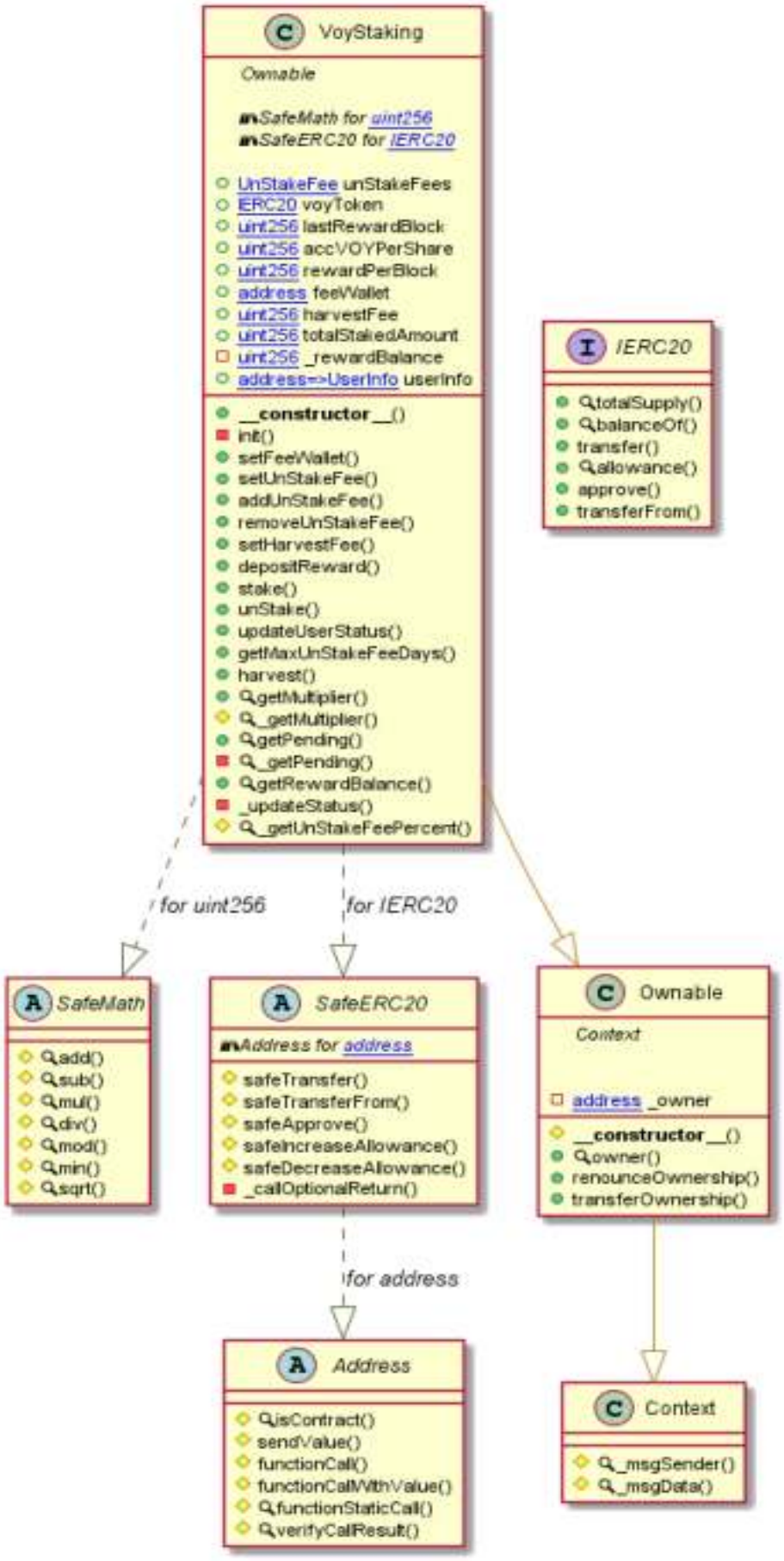
VoyXDCToken Diagram



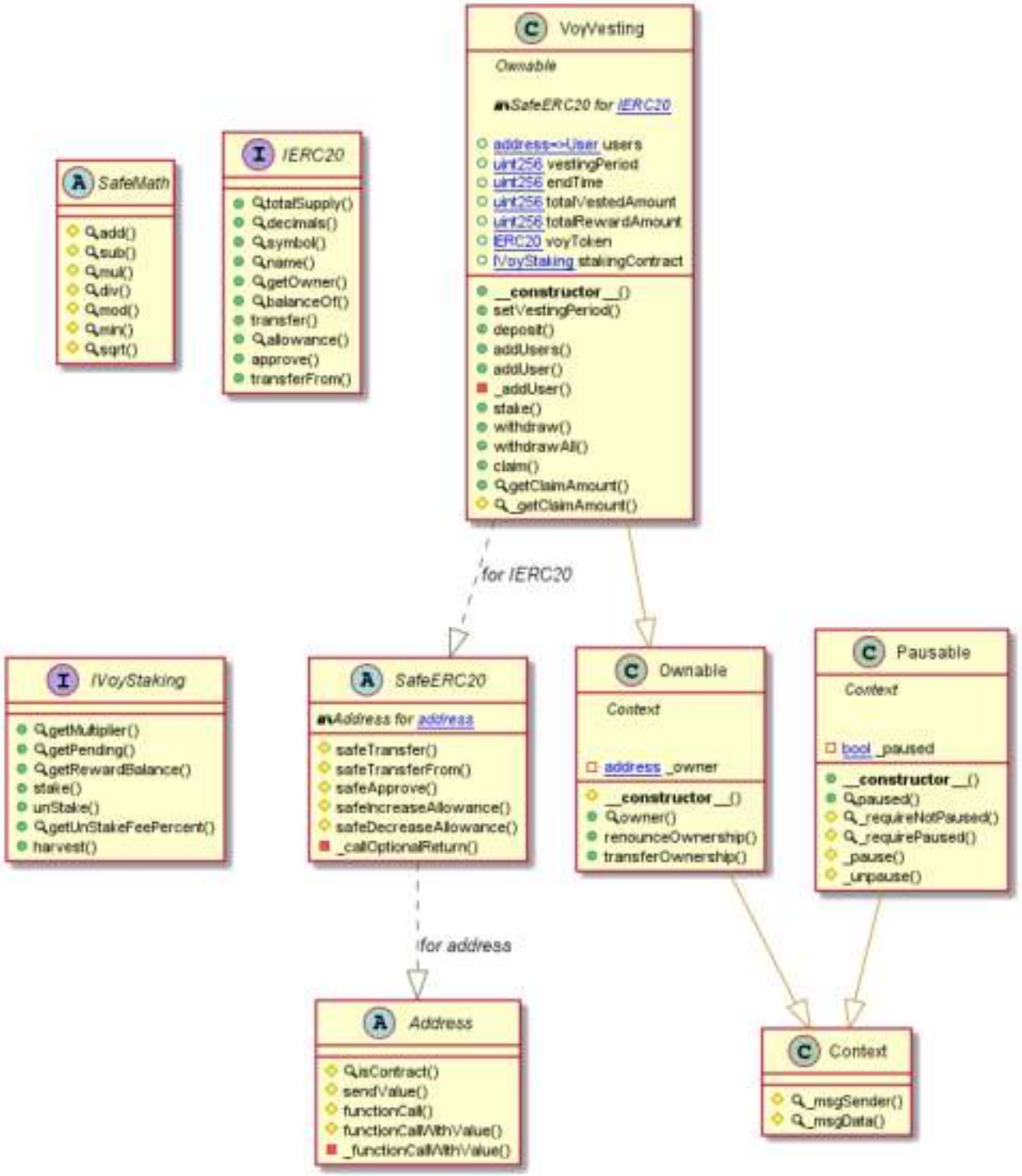
MasterChef Diagram



VoyStaking Diagram



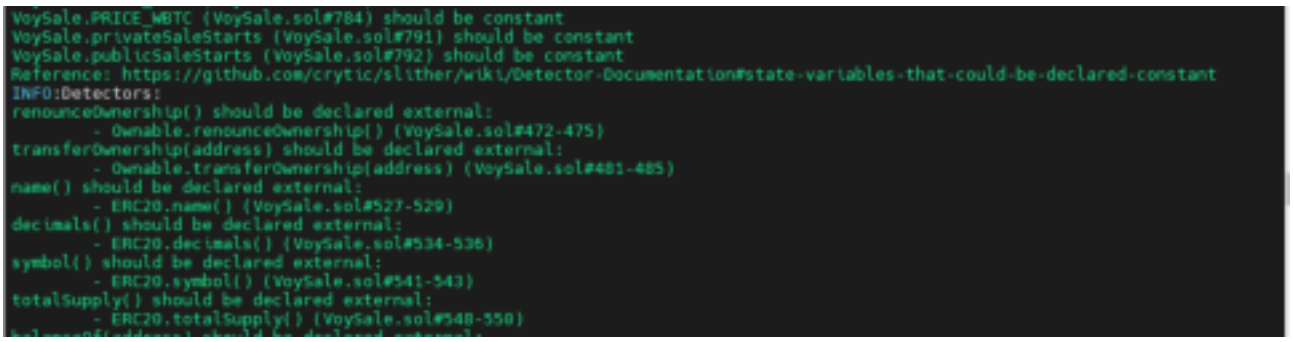
VoyVesting Diagram



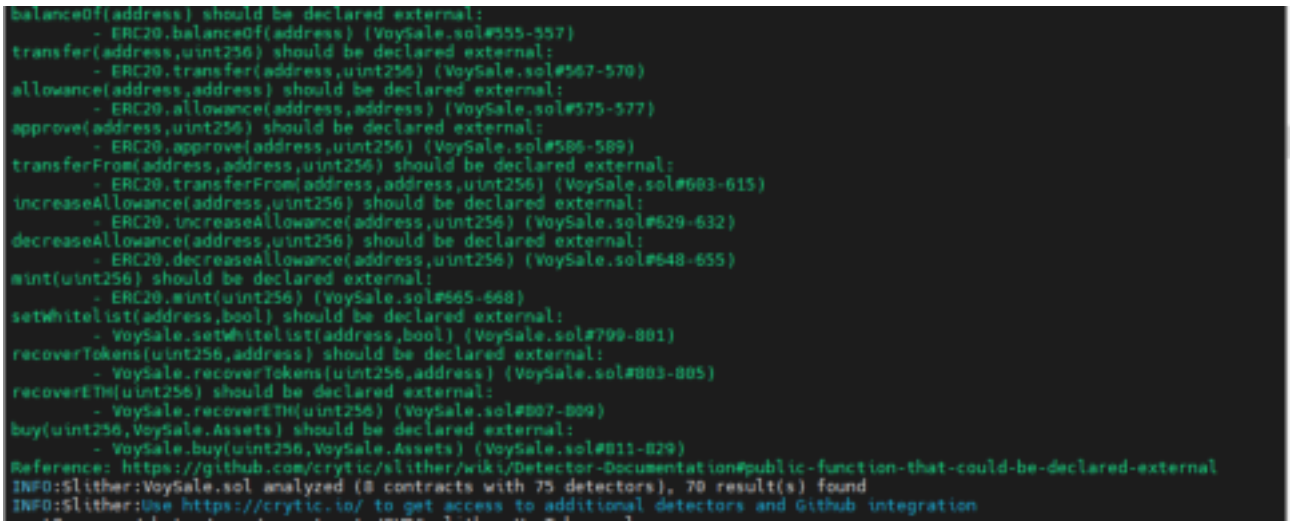
Slither Results Log

Slither log >> VoySale.sol

```
INFO:Detectors:
Redundant expression "this (VoySale.sol#211)" inContext (VoySale.sol#425-436)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
VoySale.PRICE_ETH (VoySale.sol#782) should be constant
VoySale.PRICE_USDT (VoySale.sol#783) should be constant
VoySale.PRICE_WBTC (VoySale.sol#784) should be constant
VoySale.privateSaleStarts (VoySale.sol#791) should be constant
VoySale.publicSaleStarts (VoySale.sol#792) should be constant
```

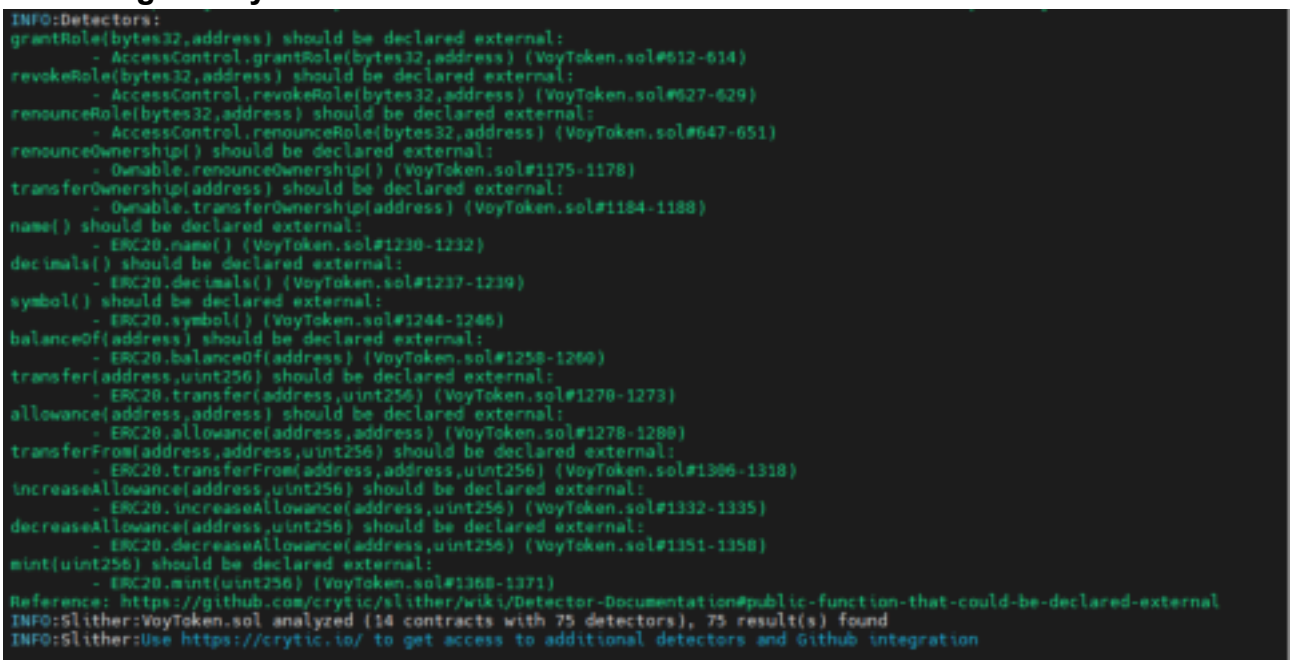


```
VoySale.PRICE_WBTC (VoySale.sol#784) should be constant
VoySale.privateSaleStarts (VoySale.sol#791) should be constant
VoySale.publicSaleStarts (VoySale.sol#792) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (VoySale.sol#472-475)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (VoySale.sol#481-485)
name() should be declared external:
        - ERC20.name() (VoySale.sol#527-529)
decimals() should be declared external:
        - ERC20.decimals() (VoySale.sol#534-536)
symbol() should be declared external:
        - ERC20.symbol() (VoySale.sol#541-543)
totalSupply() should be declared external:
        - ERC20.totalSupply() (VoySale.sol#548-550)
```



```
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (VoySale.sol#555-557)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (VoySale.sol#567-570)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (VoySale.sol#575-577)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (VoySale.sol#586-589)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (VoySale.sol#603-615)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (VoySale.sol#629-632)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (VoySale.sol#648-655)
mint(uint256) should be declared external:
        - ERC20.mint(uint256) (VoySale.sol#665-668)
setWhitelist(address,bool) should be declared external:
        - VoySale.setWhitelist(address,bool) (VoySale.sol#799-801)
recoverTokens(uint256,address) should be declared external:
        - VoySale.recoverTokens(uint256,address) (VoySale.sol#803-805)
recoverETH(uint256) should be declared external:
        - VoySale.recoverETH(uint256) (VoySale.sol#807-809)
buy(uint256,VoySale.Assets) should be declared external:
        - VoySale.buy(uint256,VoySale.Assets) (VoySale.sol#811-829)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:VoySale.sol analyzed (8 contracts with 75 detectors), 70 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> VoyToken.sol



```
INFO:Detectors:
grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (VoyToken.sol#612-614)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (VoyToken.sol#627-629)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (VoyToken.sol#647-651)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (VoyToken.sol#1175-1178)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (VoyToken.sol#1184-1188)
name() should be declared external:
        - ERC20.name() (VoyToken.sol#1230-1232)
decimals() should be declared external:
        - ERC20.decimals() (VoyToken.sol#1237-1239)
symbol() should be declared external:
        - ERC20.symbol() (VoyToken.sol#1244-1246)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (VoyToken.sol#1258-1260)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (VoyToken.sol#1270-1273)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (VoyToken.sol#1278-1280)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (VoyToken.sol#1306-1318)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (VoyToken.sol#1332-1335)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (VoyToken.sol#1351-1358)
mint(uint256) should be declared external:
        - ERC20.mint(uint256) (VoyToken.sol#1368-1371)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:VoyToken.sol analyzed (14 contracts with 75 detectors), 75 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> XDCBridge.sol

```
INFO:Detectors:
Parameter XDCBridge.initiateSwap(address,uint256)._user (XDCBridge.sol#879) is not in mixedCase
Parameter XDCBridge.initiateSwap(address,uint256)._amount (XDCBridge.sol#879) is not in mixedCase
Parameter XDCBridge.returnSwap(address,uint256,string)._user (XDCBridge.sol#886) is not in mixedCase
Parameter XDCBridge.returnSwap(address,uint256,string)._amount (XDCBridge.sol#886) is not in mixedCase
Parameter XDCBridge.returnSwap(address,uint256,string)._xdcTxHash (XDCBridge.sol#886) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (XDCBridge.sol#432)" inContext (XDCBridge.sol#426-435)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (XDCBridge.sol#562-565)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (XDCBridge.sol#571-575)
name() should be declared external:
        - ERC20.name() (XDCBridge.sol#617-619)
decimals() should be declared external:
        - ERC20.decimals() (XDCBridge.sol#624-626)
symbol() should be declared external:
        - ERC20.symbol() (XDCBridge.sol#631-633)
totalSupply() should be declared external:
        - ERC20.totalSupply() (XDCBridge.sol#638-640)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (XDCBridge.sol#645-647)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (XDCBridge.sol#657-660)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (XDCBridge.sol#665-667)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (XDCBridge.sol#676-679)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (XDCBridge.sol#693-705)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (XDCBridge.sol#719-722)
```

```
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (XDCBridge.sol#719-722)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (XDCBridge.sol#738-745)
mint(uint256) should be declared external:
        - ERC20.mint(uint256) (XDCBridge.sol#755-758)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:XDCBridge.sol analyzed (9 contracts with 75 detectors), 51 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> VoyXDCToken.sol

```
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (VoyXDCToken.sol#472-475)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (VoyXDCToken.sol#481-485)
name() should be declared external:
        - ERC20.name() (VoyXDCToken.sol#527-529)
decimals() should be declared external:
        - ERC20.decimals() (VoyXDCToken.sol#534-536)
symbol() should be declared external:
        - ERC20.symbol() (VoyXDCToken.sol#541-543)
totalSupply() should be declared external:
        - ERC20.totalSupply() (VoyXDCToken.sol#548-550)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (VoyXDCToken.sol#555-557)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (VoyXDCToken.sol#567-570)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (VoyXDCToken.sol#575-577)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (VoyXDCToken.sol#586-589)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (VoyXDCToken.sol#603-615)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (VoyXDCToken.sol#629-632)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (VoyXDCToken.sol#648-655)
mint(uint256) should be declared external:
        - ERC20.mint(uint256) (VoyXDCToken.sol#665-668)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:VoyXDCToken.sol analyzed (7 contracts with 75 detectors), 40 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> MasterChef.sol

```
INFO:Detectors:
Redundant expression "this (MasterChef.sol#556)" inContext (MasterChef.sol#550-559)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (MasterChef.sol#597-600)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (MasterChef.sol#606-610)
name() should be declared external:
        - ERC20.name() (MasterChef.sol#653-655)
decimals() should be declared external:
        - ERC20.decimals() (MasterChef.sol#660-662)
symbol() should be declared external:
        - ERC20.symbol() (MasterChef.sol#667-669)
balanceOf(address) should be declared external:
```

```
mint(uint256) should be declared external:
        - ERC20.mint(uint256) (MasterChef.sol#791-794)
mint(address,uint256) should be declared external:
        - VoyToken.mint(address,uint256) (MasterChef.sol#943-947)
burn(address,uint256) should be declared external:
        - VoyToken.burn(address,uint256) (MasterChef.sol#949-951)
add(uint256,IERC20,bool) should be declared external:
        - MasterChef.add(uint256,IERC20,bool) (MasterChef.sol#1044-1056)
set(uint256,uint256,bool) should be declared external:
        - MasterChef.set(uint256,uint256,bool) (MasterChef.sol#1059-1065)
setMigrator(IMigratorChef) should be declared external:
        - MasterChef.setMigrator(IMigratorChef) (MasterChef.sol#1068-1070)
migrate(uint256) should be declared external:
        - MasterChef.migrate(uint256) (MasterChef.sol#1073-1082)
deposit(uint256,uint256) should be declared external:
        - MasterChef.deposit(uint256,uint256) (MasterChef.sol#1136-1148)
withdraw(uint256,uint256) should be declared external:
        - MasterChef.withdraw(uint256,uint256) (MasterChef.sol#1151-1162)
emergencyWithdraw(uint256) should be declared external:
        - MasterChef.emergencyWithdraw(uint256) (MasterChef.sol#1165-1172)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MasterChef.sol analyzed (10 contracts with 75 detectors), 83 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> VoyStaking.sol

```
INFO:Detectors:
Variable VoyStaking.init().unStakeFee1 (VoyStaking.sol#671-674) is too similar to VoyStaking.init().unStakeFee2 (VoyStaking.so
l#677-680)
Variable VoyStaking.init().unStakeFee2 (VoyStaking.sol#677-680) is too similar to VoyStaking.init().unStakeFee3 (VoyStaking.so
l#683-686)
Variable VoyStaking.init().unStakeFee2 (VoyStaking.sol#677-680) is too similar to VoyStaking.init().unStakeFee4 (VoyStaking.so
l#689-692)
Variable VoyStaking.init().unStakeFee2 (VoyStaking.sol#677-680) is too similar to VoyStaking.unStakeFees (VoyStaking.sol#633)
Variable VoyStaking.init().unStakeFee1 (VoyStaking.sol#671-674) is too similar to VoyStaking.init().unStakeFee3 (VoyStaking.so
l#683-686)
Variable VoyStaking.init().unStakeFee3 (VoyStaking.sol#683-686) is too similar to VoyStaking.init().unStakeFee4 (VoyStaking.so
l#689-692)
Variable VoyStaking.init().unStakeFee3 (VoyStaking.sol#683-686) is too similar to VoyStaking.unStakeFees (VoyStaking.sol#633)
Variable VoyStaking.init().unStakeFee1 (VoyStaking.sol#671-674) is too similar to VoyStaking.init().unStakeFee4 (VoyStaking.so
l#689-692)
Variable VoyStaking.init().unStakeFee4 (VoyStaking.sol#689-692) is too similar to VoyStaking.unStakeFees (VoyStaking.sol#633)
Variable VoyStaking.init().unStakeFee1 (VoyStaking.sol#671-674) is too similar to VoyStaking.unStakeFees (VoyStaking.sol#633)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (VoyStaking.sol#599-602)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (VoyStaking.sol#608-612)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:VoyStaking.sol analyzed (7 contracts with 75 detectors), 59 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> VoyVesting.sol

```
INFO:Detectors:
Pragma version^0.8.0 (VoyVesting.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
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) (VoyVesting.sol#224-230):
        - (success) = recipient.call{value: amount}() (VoyVesting.sol#228)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (VoyVesting.sol#303-329):
        - (success,returndata) = target.call{value: weiValue}(data) (VoyVesting.sol#312)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter VoyVesting.setVestingPeriod(uint56)._vestingPeriod (VoyVesting.sol#735) is not in mixedCase
Parameter VoyVesting.deposit(uint256)._amount (VoyVesting.sol#741) is not in mixedCase
Parameter VoyVesting.addUsers(address[],uint256[])._userAddresses (VoyVesting.sol#745) is not in mixedCase
Parameter VoyVesting.addUsers(address[],uint256[])._amounts (VoyVesting.sol#745) is not in mixedCase
Parameter VoyVesting.addUser(address,uint256)._userAddress (VoyVesting.sol#752) is not in mixedCase
Parameter VoyVesting.addUser(address,uint256)._amount (VoyVesting.sol#752) is not in mixedCase
Parameter VoyVesting.withdraw(uint256)._amount (VoyVesting.sol#772) is not in mixedCase
Parameter VoyVesting.getClaimAmount(address)._userAddress (VoyVesting.sol#800) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (VoyVesting.sol#515)" inContext (VoyVesting.sol#509-518)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (VoyVesting.sol#550-550)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (VoyVesting.sol#565-569)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:VoyVesting.sol analyzed (9 contracts with 75 detectors), 49 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

VoySale.sol

Security

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 44:16:

Gas & Economy

Gas costs:

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

Miscellaneous

Constant/View/Pure functions:

IVesting.initiateVesting(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 8:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 44:8:

VoyToken.sol

Security

Inline assembly:

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

more

Pos: 45:8:

Gas & Economy

Gas costs:

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

Pos: 27:4:

Gas costs:

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

Pos: 33:4:

Miscellaneous

Constant/View/Pure functions:

VoyToken.isContract(address) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 43:4:

Similar variable names:

VoyToken.swap(address,uint256) : Variables have very similar names "bridges" and "_bridge". Note: Modifiers are currently not considered by this static analysis.

Pos: 51:19:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 55:8:

XDCBridge.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in XDCBridge.initiateSwap(address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 20:4:

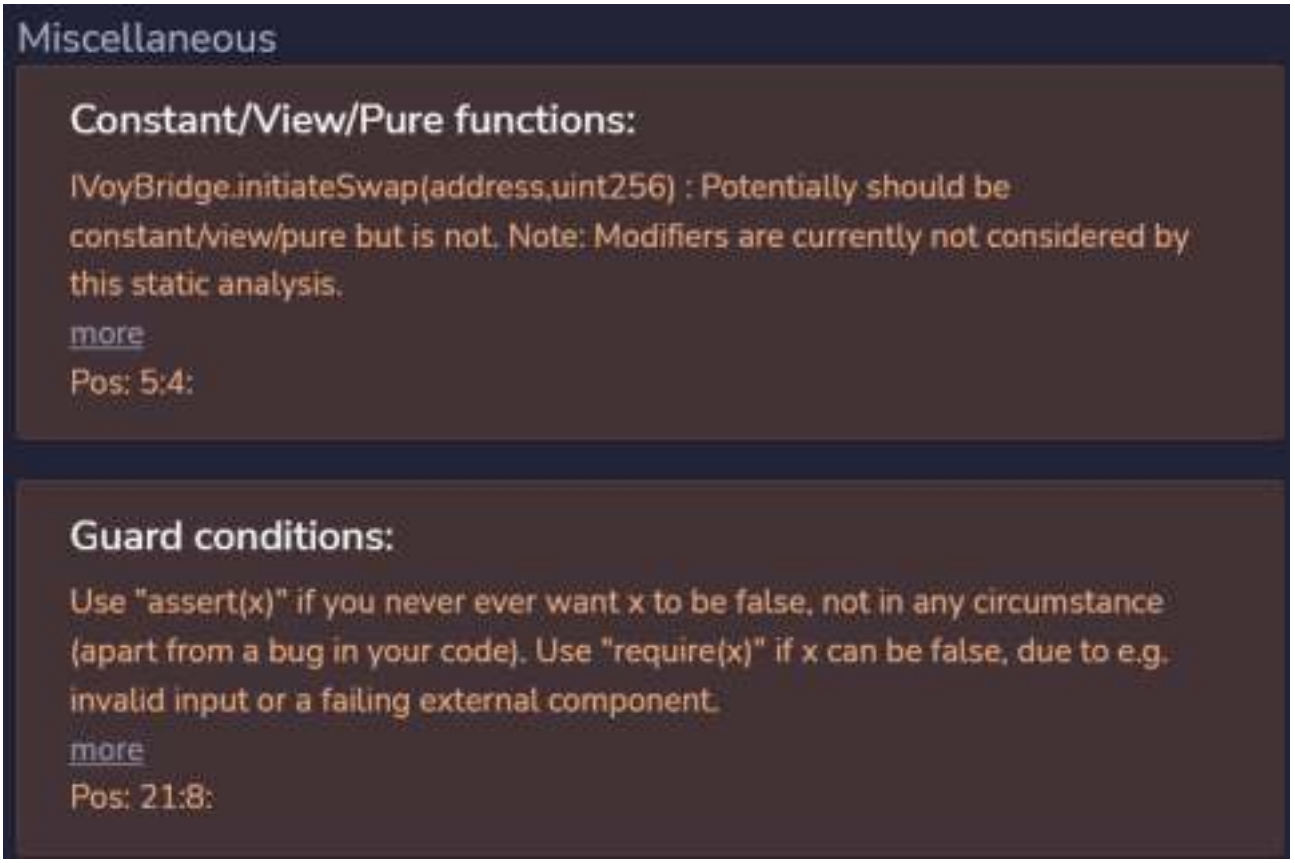
Gas & Economy

Gas costs:

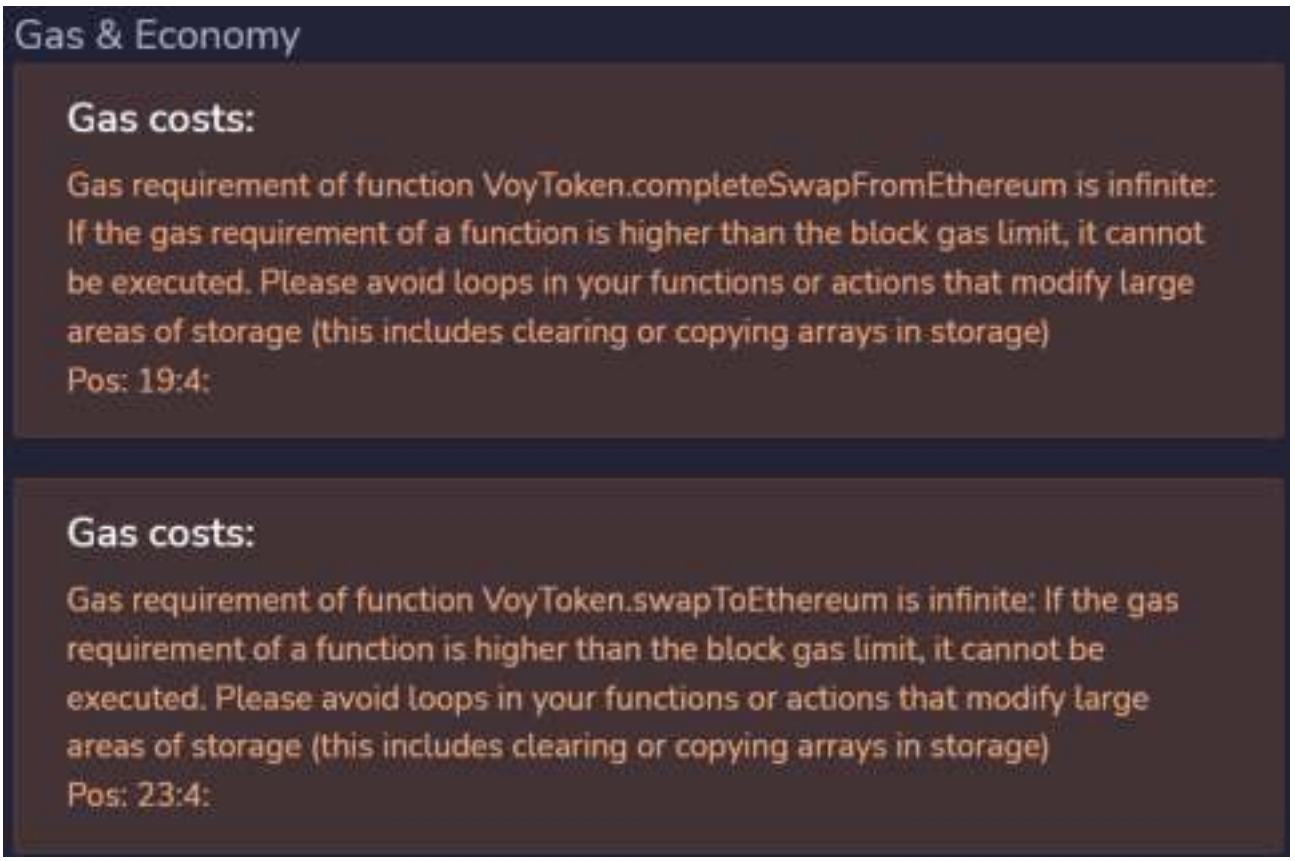
Gas requirement of function XDCBridge.initiateSwap is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 20:4:

Gas costs:

Gas requirement of function XDCBridge.returnSwap is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 27:4:



VoyXDCToken.sol



MasterChef.sol

Gas & Economy

Gas costs:

Gas requirement of function MasterChef.set is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 116:4:

Miscellaneous

Constant/View/Pure functions:

IMigratorChef.migrate(contract IERC20) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 21:4:

Similar variable names:

VoyToken.removeAdmin(address) : Variables have very similar names "admins" and "_admin". Note: Modifiers are currently not considered by this static analysis.
Pos: 45:39:

No return:

IMigratorChef.migrate(contract IERC20): Defines a return type but never explicitly returns a value.
Pos: 21:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 50:8:

VoyStaking.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in VoyStaking.unStake(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 172:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 341:58:

Gas & Economy

Gas costs:

Gas requirement of function VoyStaking.voyToken is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 31:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 134:8:

Miscellaneous

Constant/View/Pure functions:

VoyStaking.getMaxUnStakeFeeDays() : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 243:4:

Similar variable names:

VoyStaking.unStake(uint256) : Variables have very similar names "unStakeFee" and "unStakeFees". Note: Modifiers are currently not considered by this static analysis.
Pos: 207:41:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 179:8:

VoyVesting.sol

Security

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 149:22:

Gas & Economy

Gas costs:

Gas requirement of function VoyVesting.claim is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 117:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 81:8:

Miscellaneous

Constant/View/Pure functions:

VoyVesting.withdrawAll() : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 111:4:

Similar variable names:

VoyVesting._getClaimAmount(address) : Variables have very similar names "user" and "users". Note: Modifiers are currently not considered by this static analysis.
Pos: 147:56:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 99:8:

Solhint Linter

VoySale.sol

```
VoySale.sol:2:1: Error: Compiler version >=0.8.0 does not satisfy the
r semver requirement
VoySale.sol:24:5: Error: Explicitly mark visibility of state
VoySale.sol:26:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
VoySale.sol:44:17: Error: Avoid to make time-based decisions in your
business logic
VoySale.sol:44:56: Error: Avoid to make time-based decisions in your
business logic
```

VoyToken.sol

```
VoyToken.sol:2:1: Error: Compiler version >=0.8.0 does not satisfy
the r semver requirement
VoyToken.sol:17:20: Error: Variable name must be in mixedCase
VoyToken.sol:20:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
VoyToken.sol:45:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
```

XDCBridge.sol

```
XDCBridge.sol:2:1: Error: Compiler version ^0.8.0 does not satisfy
the r semver requirement
XDCBridge.sol:11:5: Error: Explicitly mark visibility of state
XDCBridge.sol:16:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
```

VoyXDCToken.sol

```
VoyXDCToken.sol:2:1: Error: Compiler version >=0.8.0 does not satisfy
the r semver requirement
VoyXDCToken.sol:15:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
```

MasterChef.sol

```
MasterChef.sol:2:1: Error: Compiler version >=0.8.0 does not satisfy
the r semver requirement
MasterChef.sol:83:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
```

VoyStaking.sol

```
VoyStaking.sol:3:1: Error: Compiler version ^0.8.0 does not satisfy
the r semver requirement
VoyStaking.sol:55:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
VoyStaking.sol:341:59: Error: Avoid to make time-based decisions in
your business logic
```

VoyVesting.sol

```
VoyVesting.sol:3:1: Error: Compiler version ^0.8.0 does not satisfy
the r semver requirement
VoyVesting.sol:44:17: Error: Avoid to make time-based decisions in
your business logic
VoyVesting.sol:58:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
VoyVesting.sol:102:19: Error: Avoid to make time-based decisions in
your business logic
VoyVesting.sol:149:23: Error: Avoid to make time-based decisions in
your business logic
```

Software analysis result:

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

Email: audit@EtherAuthority.io