Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: Angry Sheep Club
Website: https://angrysheep.club
Platform: Ethereum
Language: Solidity
Date: March 22nd, 2023

Table of contents

Email: audit@EtherAuthority.io

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

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by Angry Sheep Club to perform the Security audit of the Angry Sheep Club 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 March 22nd, 2023.

The purpose of this audit was to address the following:

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

Project Background

- Angry Sheep Club is a Private Membership Club that provides tangible benefits for its members.
- Angry Sheep Club is allowed a maximum of 5 NFT to be minted per whitelist member.
- There are 5 smart contracts, which were included in the audit scope. And there were some standard library code such as OpenZepelin, which were excluded. Because those standard library code is considered as time tested and community audited, so we can safely ignore them.

Email: audit@EtherAuthority.io

Audit scope

Name	Code Review and Security Analysis Report for Angry Sheep Club Smart Contracts
Platform	**Ethereum / Solidity**
File 1	First100Pool.sol
File 1 MD5 Hash	B959C91C3BA1A670AF3E12684B733499
File 2	HonoraryMembersPool.sol
File 2 MD5 Hash	9FE0A96E7D9647F2A8E93183BFBD0918
File 3	OGMintersPool.sol
File 3 MD5 Hash	F1F156F61698A57A5DE51F2FAAC98A30
File 4	AngrySheepDistribution.sol
File 4 MD5 Hash	124FAC0EB355785AD9F0D90DB695FE2C
File 5	AngrySheepClub.sol
File 5 MD5 Hash	8843499DFB361813AE49E5818BF8A986
Audit Date	March 22nd, 2023
Revised Date	April 4th, 2023

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 First100Pool.sol **Owner Specifications:** • The Owner can flip claim rewards. • Merkle root value can be set by the owner.	**YES, This is valid.**
File 2 HonoraryMembersPool.sol • Name: Angry Sheep Club: Honorary Collection • Symbol: ASCHM • Total Supply: 50 **Owner Specifications:** • The Owner can flip contract deposits. • BaseURI can be set by the owner. • Withdraw money by the owner.	**YES, This is valid.**
File 3 OGMintersPool.sol **Owner Specifications:** • The Owner can flip claim rewards. • Merkle root value can be set by the owner.	**YES, This is valid.**
File 4 AngrySheepDistribution.sol • Name: AngrySheepClub-Distribution • Symbol: ascd • Pre Sale Amount: 100 • Sub Pre Sale Amount: 25 • Total Supply: 15001 **Owner Specifications:** • OG Minter spool address can be set by the owner. • Distribution Merkle root value can be set by the	**YES, This is valid.**

owner. • The Owner can flip claim rewards. • Distribute money by the owner.	
File 5 AngrySheepClub.sol • Name: AngrySheepClub: Platinum Series • Symbol: ASC • Presale Size: 2000 • Sub Presale Amount: 25 • Total Supply: 15000 • Price: 0.0001 ether • Maximum amount that user can mint per transaction: 5 • Mint Limit: 5 **Owner Specifications:** • The owner can send the nft number 10k-15k to a wallet address. • The owner can send the nft number 10k-15k to a wallet address. • The owner can mint unlimited tokens. • The owner can presale minting tokens. • Withdraw money by the owner. • Update mint price by the owner. • Maximum transaction value can be set by the owner. • Maximum transaction value can be set by the owner.	**YES, This is valid.**

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, 0 high, 0 medium and 0 low and some very low level issues in the revised smart contract code.

Email: audit@EtherAuthority.io

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	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: PASSED

Code Quality

This audit scope has 5 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in the Angry Sheep Club 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 Angry Sheep Club Protocol.

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

Code parts are not well commented on smart contracts.

Documentation

We were given an Angry Sheep Club Protocol smart contract code in the form of a goerli.etherscan.io weblink. The hash of that code is mentioned above in the table.

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

Another source of information was its official website: https://angrysheep.club which provided rich information about the project architecture and tokenomics.

Use of Dependencies

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

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

AS-IS overview

First100Pool.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	_transferOwnership	internal	Passed	No Issue
7	flipclaimReward	write	access only Owner	No Issue
8	deposit	write	Passed	No Issue
9	claim_First100_RewardM	write	Passed	No Issue
10	setMerkleRoot	write	access only Owner	No Issue
11	getMerkleRoot	read	Passed	No Issue

HonoraryMembersPool.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	_transferOwnership	internal	Passed	No Issue
7	totalSupply	read	Passed	No Issue
8	tokenByIndex	read	Passed	No Issue
9	tokenOfOwnerByIndex	read	Passed	No Issue
10	supportsInterface	read	Passed	No Issue
11	balanceOf	read	Passed	No Issue
12	_numberMinted	internal	Passed	No Issue
13	ownershipOf	internal	Passed	No Issue
14	ownerOf	read	Passed	No Issue
15	name	read	Passed	No Issue
16	symbol	read	Passed	No Issue
17	tokenURI	read	Passed	No Issue
18	_baseURI	internal	Passed	No Issue
19	_getUriExtension	internal	Passed	No Issue
20	approve	write	Passed	No Issue
21	getApproved	read	Passed	No Issue
22	setApprovalForAll	write	Passed	No Issue
23	isApprovedForAll	read	Passed	No Issue

24	transferFrom	write	Passed	No Issue
25	safeTransferFrom	write	Passed	No Issue
26	safeTransferFrom	write	Passed	No Issue
27	_exists	internal	Passed	No Issue
28	_safeMint	internal	Passed	No Issue
29	_safeMint	internal	Passed	No Issue
30	_transfer	write	Passed	No Issue
31	_approve	write	Passed	No Issue
32	_setOwnersExplicit	internal	Passed	No Issue
33	_checkOnERC721Received	write	Passed	No Issue
34	_beforeTokenTransfers	internal	Passed	No Issue
35	_afterTokenTransfers	internal	Passed	No Issue
36	nonReentrant	modifier	Passed	No Issue
37	setApprovalForAll	write	access only Allowed Operator Approval	No Issue
38	approve	write	access only Allowed Operator Approval	No Issue
39	transferFrom	write	access only Allowed Operator	No Issue
40	safeTransferFrom	write	access only Allowed Operator	No Issue
41	safeTransferFrom	write	access only Allowed Operator	No Issue
42	flipcontractDeposit	write	access only Owner	No Issue
43	flipclaimReward	write	access only Owner	No Issue
44	deposit	write	Passed	No Issue
45	claim_HonoraryMembers_Reward	write	Passed	No Issue
46	tokenURI	read	Passed	No Issue
47	setBaseURI	external	access only Owner	No Issue
48	_baseURI	internal	Passed	No Issue
49	numberMinted	read	Passed	No Issue
50	getOwnershipData	external	Passed	No Issue
51	withdrawMoney	external	Passed	No Issue
52	callerIsUser	modifier	Passed	No Issue

OGMintersPool.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	_transferOwnership	internal	Passed	No Issue

7	flipclaimReward	write	access only Owner	No Issue
8	deposit	write	Passed	No Issue
9	claim_OGMinters_RewardM	write	Passed	No Issue
10	claim_OGMinters_Reward	write	Passed	No Issue
11	setMerkleRoot	write	access only Owner	No Issue
12	getMerkleRoot	read	Passed	No Issue

AngrySheepDistribution.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	_transferOwnership	internal	Passed	No Issue
7	totalSupply	read	Passed	No Issue
8	tokenByIndex	read	Passed	No Issue
9	tokenOfOwnerByIndex	read	Passed	No Issue
10	supportsInterface	read	Passed	No Issue
11	balanceOf	read	Passed	No Issue
12	_numberMinted	internal	Passed	No Issue
13	ownershipOf	internal	Passed	No Issue
14	ownerOf	read	Passed	No Issue
15	name	read	Passed	No Issue
16	symbol	read	Passed	No Issue
17	tokenURI	read	Passed	No Issue
18	_baseURI	internal	Passed	No Issue
19	_getUriExtension	internal	Passed	No Issue
20	approve	write	Passed	No Issue
21	getApproved	read	Passed	No Issue
22	setApprovalForAll	write	Passed	No Issue
23	isApprovedForAll	read	Passed	No Issue
24	transferFrom	write	Passed	No Issue
25	safeTransferFrom	write	Passed	No Issue
26	safeTransferFrom	write	Passed	No Issue
27	_exists	internal	Passed	No Issue
28	_safeMint	internal	Passed	No Issue
29	_safeMint	internal	Passed	No Issue
30	_transfer	write	Passed	No Issue
31	_approve	write	Passed	No Issue
32	_setOwnersExplicit	internal	Passed	No Issue
33	_checkOnERC721Received	write	Passed	No Issue
34	_beforeTokenTransfers	internal	Passed	No Issue
35	_afterTokenTransfers	internal	Passed	No Issue

36	nonReentrant	modifier	Passed	No Issue
37	flipclaimReward	write	access only Owner	No Issue
38	deposit	write	Passed	No Issue
39	claimHolderMemberRewardM	write	Passed	No Issue
40	claimHolderMemberReward	write	Passed	No Issue
41	distributeMoney	external	Passed	No Issue
42	sendToWallet	write	Passed	No Issue
43	setOGMinterspoolAddress	write	access only Owner	No Issue
44	setDistributionMerkleRoot	write	access only Owner	No Issue
45	getDistributionMerkleRoot	read	Passed	No Issue

AngrySheepClub.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	_transferOwnership	internal	Passed	No Issue
7	totalSupply	read	Passed	No Issue
8	tokenByIndex	read	Passed	No Issue
9	tokenOfOwnerByIndex	read	Passed	No Issue
10	supportsInterface	read	Passed	No Issue
11	balanceOf	read	Passed	No Issue
12	_numberMinted	internal	Passed	No Issue
13	ownershipOf	internal	Passed	No Issue
14	ownerOf	read	Passed	No Issue
15	name	read	Passed	No Issue
16	symbol	read	Passed	No Issue
17	tokenURI	read	Passed	No Issue
18	_baseURI	internal	Passed	No Issue
19	_getUriExtension	internal	Passed	No Issue
20	approve	write	Passed	No Issue
21	getApproved	read	Passed	No Issue
22	setApprovalForAll	write	Passed	No Issue
23	isApprovedForAll	read	Passed	No Issue
24	transferFrom	write	Passed	No Issue
25	safeTransferFrom	write	Passed	No Issue
26	safeTransferFrom	write	Passed	No Issue
27	_exists	internal	Passed	No Issue
28	_safeMint	internal	Passed	No Issue
29	_safeMint	internal	Passed	No Issue
30	_transfer	write	Passed	No Issue
31	_approve	write	Passed	No Issue

32	_setOwnersExplicit	internal	Passed	No Issue
33	_checkOnERC721Received	write	Passed	No Issue
34	_beforeTokenTransfers	internal	Passed	No Issue
35	_afterTokenTransfers	internal	Passed	No Issue
36	nonReentrant	modifier	Passed	No Issue
37	setApprovalForAll	write	access only Allowed Operator Approva	No Issue
38	approve	write	access only Allowed Operator Approva	No Issue
39	transferFrom	write	access only Allowed Operator	No Issue
40	safeTransferFrom	write	access only Allowed Operator	No Issue
41	safeTransferFrom	write	access only Allowed Operator	No Issue
42	sendBackup4kNFTS	external	access only Allowed Operator	No Issue
43	sendBackup1kNFTS	external	access only Allowed Operator	No Issue
44	stopMint	external	access only Owner	No Issue
45	startMint	external	access only Owner	No Issue
46	mint	external	callerIsUser	No Issue
47	PresaleMint	external	callerIsUser	No Issue
48	tokenURI	read	Passed	No Issue
49	callerIsUser	modifier	Passed	No Issue
50	setMerkleRoot	write	access only Owner	No Issue
51	getMerkleRoot	read	Passed	No Issue
52	setSubPresaleAmount	write	access only Owner	No Issue
53	getSubPresaleAmount	read	Passed	No Issue
54	setBaseURI	external	access only Owner	No Issue
55	_baseURI	internal	Passed	No Issue
56	numberMinted	read	Passed	No Issue
57	getOwnershipData	external	Passed	No Issue
58	withdrawMoney	external	access only Owner	No Issue
59	changeMintPrice	external	access only Owner	No Issue
60	changeMAX_PER_Transaction	external	access only Owner	No Issue

Severity Definitions

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

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found in the revised contract code.

High Severity

No high severity vulnerabilities were found in the revised contract code.

Medium

No medium severity vulnerabilities were found in the revised contract code.

Low

No Low severity vulnerabilities were found in the revised contract code.

Very Low / Informational / Best practices:

No Informational severity vulnerabilities were found in the revised contract code.

Email: audit@EtherAuthority.io

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:

Ownable.sol

- renounceOwnership: Deleting ownership will leave the contract without an owner, removing any owner-only functionality.
- transferOwnership: Current owner can transfer ownership of the contract to a new account.

First100Pool.sol

- flipclaimReward: The Owner can flip claim rewards.
- setMerkleRoot: Merkle root value can be set by the owner.

HonoraryMembersPool.sol

- flipcontractDeposit: The Owner can flip contract deposits.
- flipclaimReward: The Owner can flip claim rewards.
- setBaseURI: BaseURI can be set by the owner.
- withdrawMoney: Withdraw money by the owner.

OGMintersPool.sol

- flipclaimReward: The Owner can flip claim rewards.
- setMerkleRoot: Merkle root value can be set by the owner.

AngrySheepClub.sol

- sendBackup4kNFTS: The Owner can send the nft number 10k-15k to a wallet address.
- sendBackup1kNFTS: The Owner can send the nft number 10k-15k to a wallet address.
- stopMint: The Owner can stop minting.
- startMint: The Owner can start minting.

- mint: The Owner can minting tokens.
- PresaleMint: The Owner can presale minting tokens.
- setMerkleRoot: Merkle root value can be set by the owner.
- setSubPresaleAmount: Sub Presale amount can be set by the owner.
- setBaseURI: BaseURI can be set by the owner.
- withdrawMoney: Withdraw money by the owner.
- changeMintPrice: Update mint price by the owner.
- changeMAX_PER_Transaction: Maximum transaction value can be set by the owner.

AngrySheepDistribution.sol

- flipclaimReward: The Owner can flip claim rewards.
- distributeMoney: Distribute money by the owner.
- setOGMinterspoolAddress: OG Minter spool address can be set by the owner.
- setDistributionMerkleRoot: Distribution Merkle root value can be set by the owner.

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 goerli.etherscan.io weblink. And we have used all possible tests based on given objects as files. We had not observed any severity issues in the revised smart contracts. **So, the smart contracts are ready for the mainnet deployment**.

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

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

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

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

Our Methodology

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

Manual Code Review:

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

Vulnerability Analysis:

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

Documenting Results:

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

Suggested Solutions:

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

Disclaimers

EtherAuthority.io Disclaimer

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

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

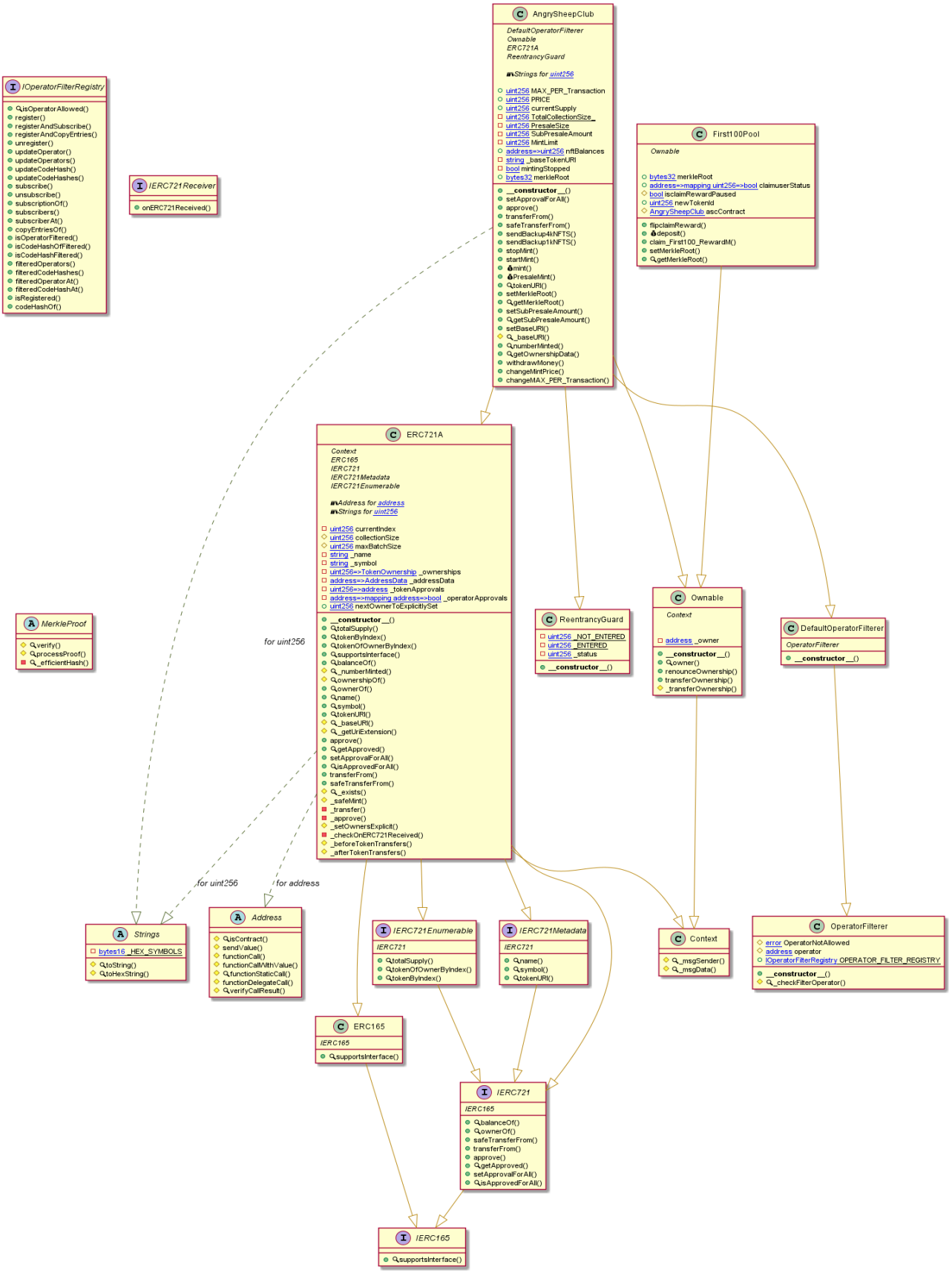
Technical Disclaimer

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

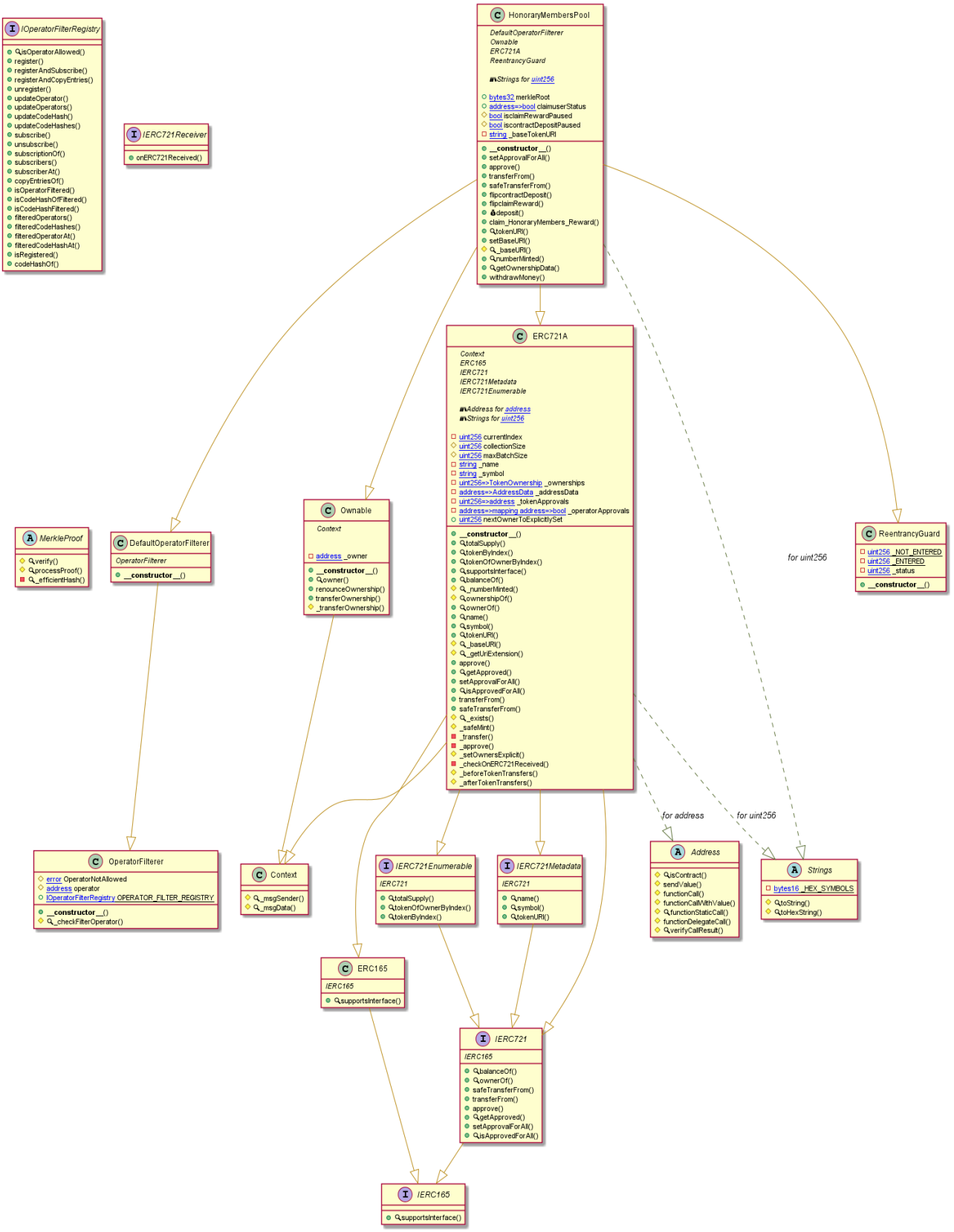
Appendix

Code Flow Diagram - Angry Sheep Club

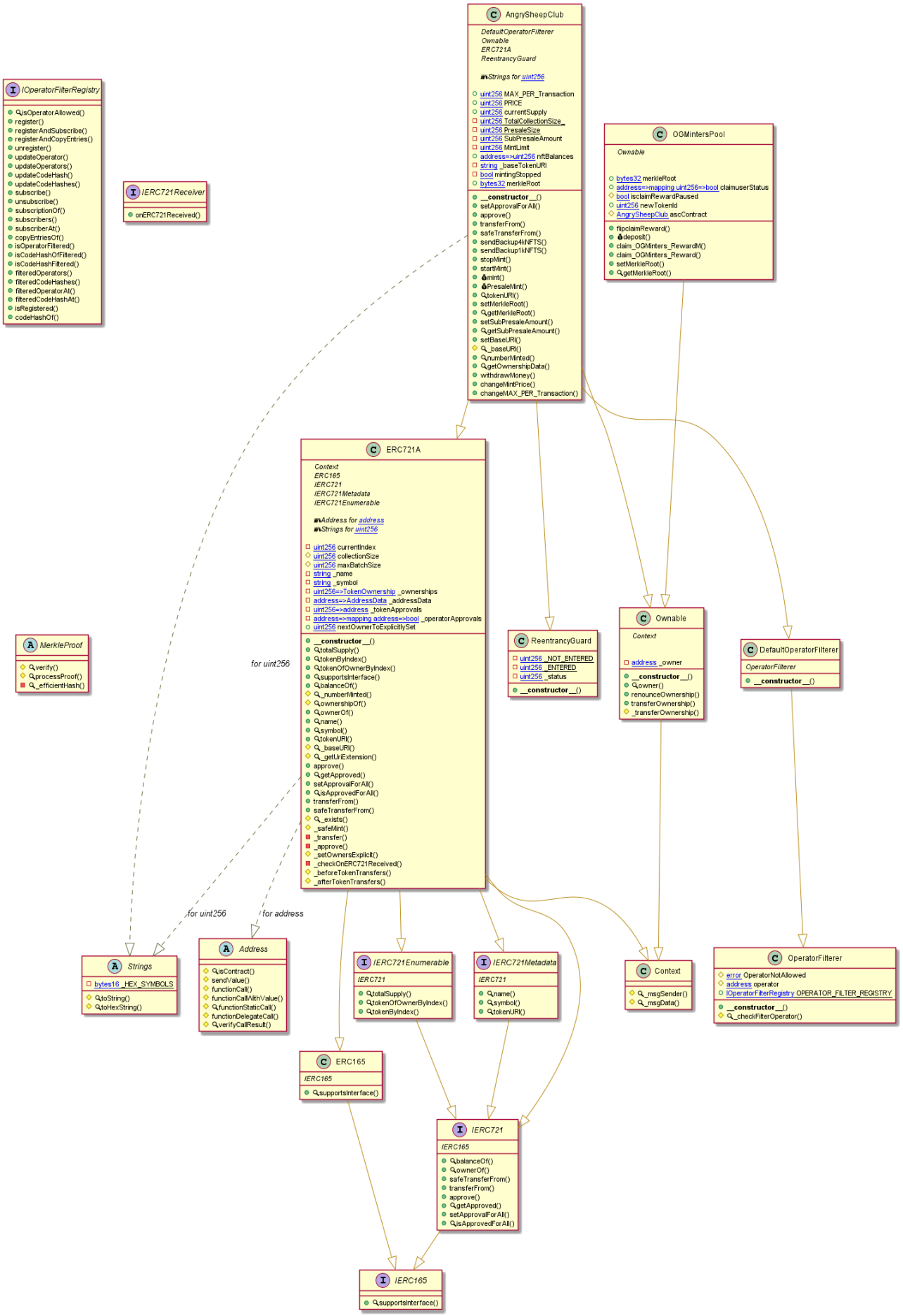
First100Pool Diagram



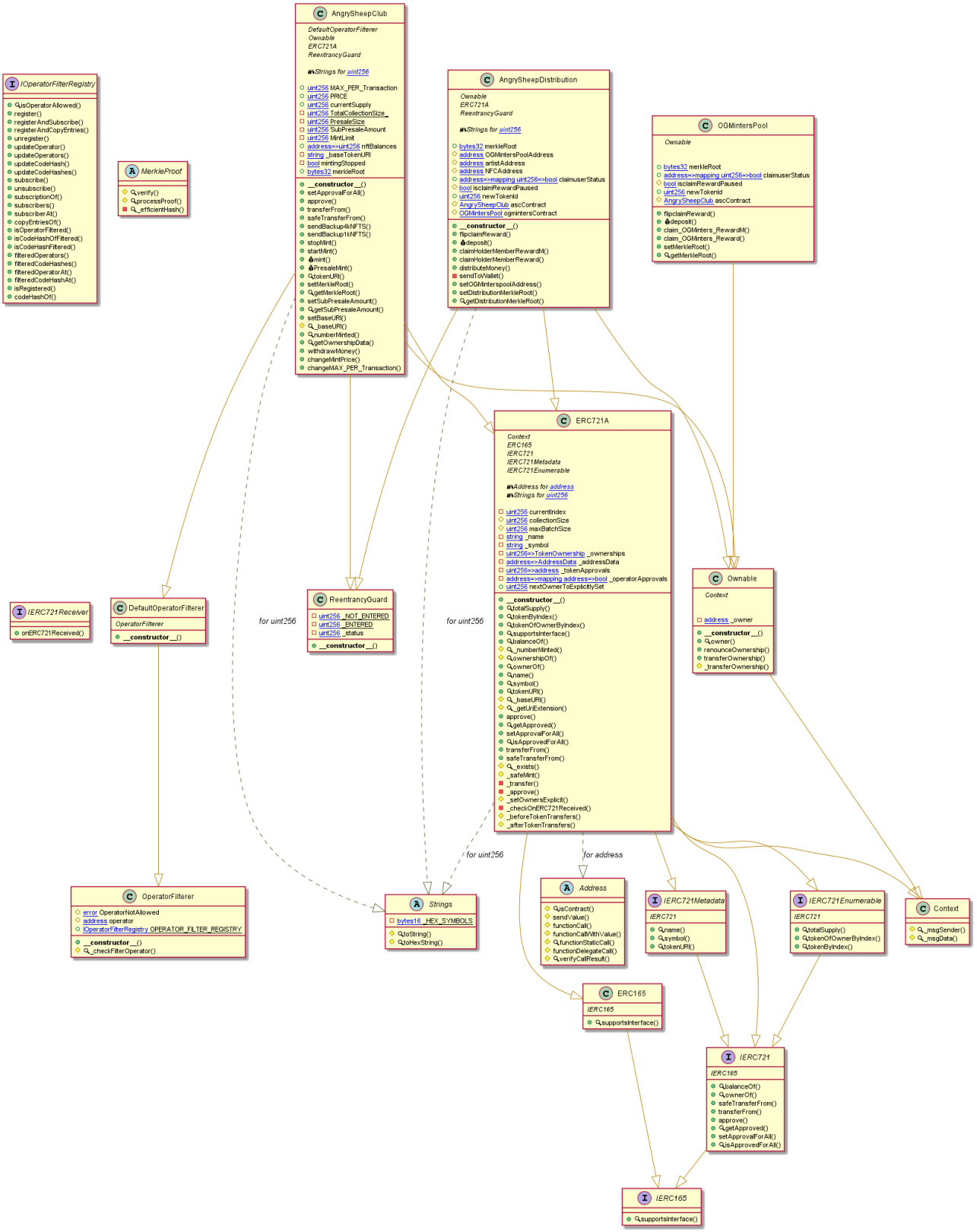
HonoraryMembersPool Diagram



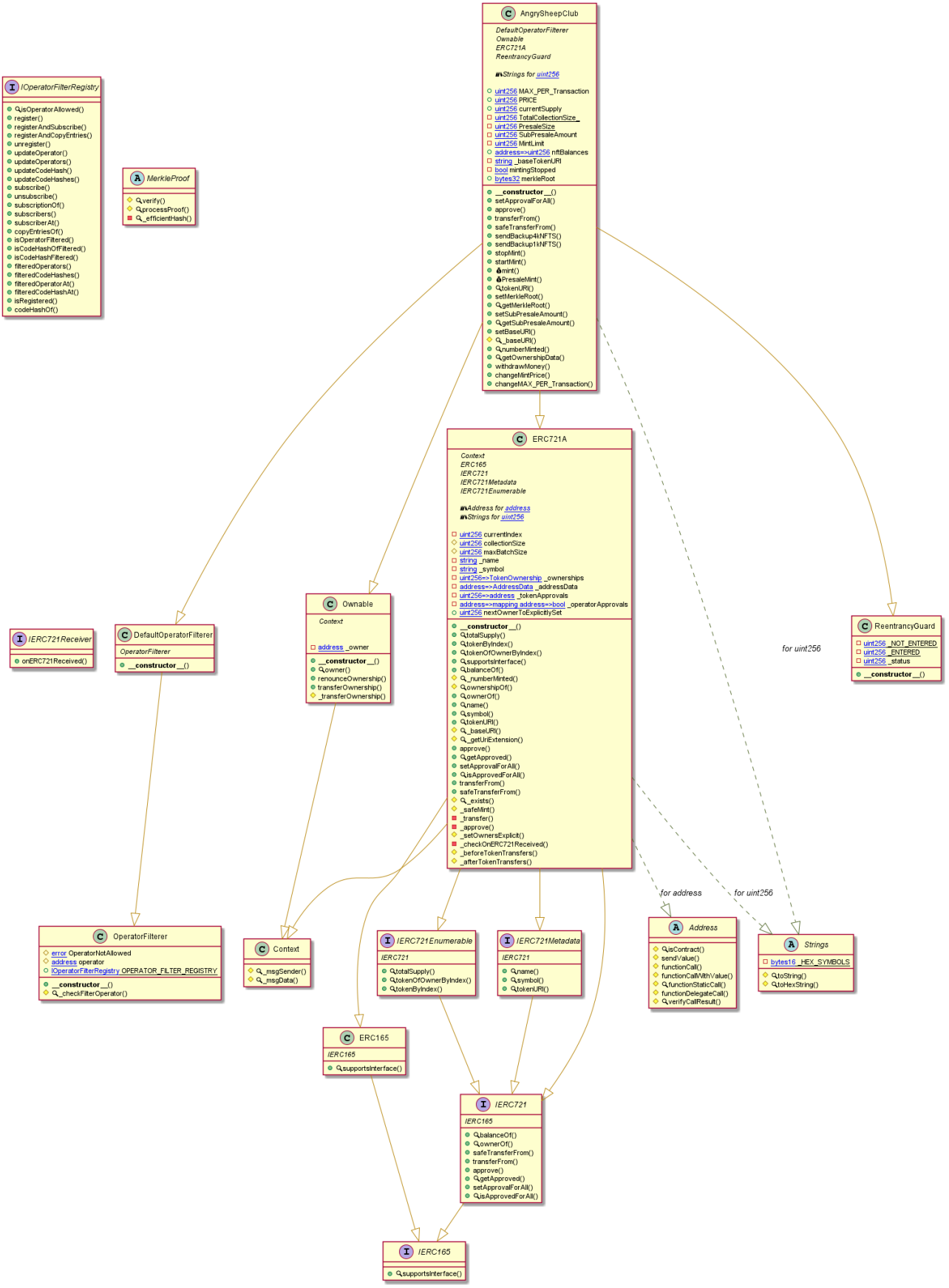
OGMintersPool Diagram



AngrySheepDistribution Diagram



AngrySheepClub Diagram



Slither Results Log

Slither log >> First100Pool.sol

```
AngrySheepClub.numberMinted(address).owner (First100Pool.sol#1076) shadows:
        - Ownable.owner() (First100Pool.sol#414-416) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

AngrySheepClub.setSubPresaleAmount(uint256) (First100Pool.sol#1062-1064) should emit an event for:
        - SubPresaleAmount = s (First100Pool.sol#1063)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Variable 'ERC721A._checkOnERC721Received(address,address,uint256,bytes).retval (First100Pool.sol#836)' in ERC721A._checkOnERC7
21Received(address,address,uint256,bytes) (First100Pool.sol#827-850) potentially used before declaration: retval == IERC721Rec
eiver(to).onERC721Received.selector (First100Pool.sol#837)
Variable 'ERC721A._checkOnERC721Received(address,address,uint256,bytes).reason (First100Pool.sol#838)' in ERC721A._checkOnERC7
21Received(address,address,uint256,bytes) (First100Pool.sol#827-850) potentially used before declaration: reason.length == 0 (
First100Pool.sol#839)
Variable 'ERC721A._checkOnERC721Received(address,address,uint256,bytes).reason (First100Pool.sol#838)' in ERC721A._checkOnERC7
21Received(address,address,uint256,bytes) (First100Pool.sol#827-850) potentially used before declaration: revert(uint256,uint2
56)(32 + reason,mload(uint256)(reason)) (First100Pool.sol#843)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
```

```
Low level call in Address.sendValue(address,uint256) (First100Pool.sol#302-307):
        - (success) = recipient.call{value: amount}() (First100Pool.sol#305)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (First100Pool.sol#328-339):
        - (success,returndata) = target.call{value: value}(data) (First100Pool.sol#337)
Low level call in Address.functionStaticCall(address,bytes,string) (First100Pool.sol#344-353):
        - (success,returndata) = target.staticcall(data) (First100Pool.sol#351)
Low level call in Address.functionDelegateCall(address,bytes,string) (First100Pool.sol#359-368):
        - (success,returndata) = target.delegatecall(data) (First100Pool.sol#366)
Low level call in AngrySheepClub.withdrawMoney() (First100Pool.sol#1087-1090):
        - (NFC) = msg.sender.call{value: address(this).balance}() (First100Pool.sol#1088)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter ERC721A.safeTransferFrom(address,address,uint256,bytes)._data (First100Pool.sol#709) is not in mixedCase
Function AngrySheepClub.PresaleMint(uint256,bytes32[]) (First100Pool.sol#1016-1038) is not in mixedCase
Parameter AngrySheepClub.changeMintPrice(uint256)._newPrice (First100Pool.sol#1092) is not in mixedCase
Function AngrySheepClub.changeMAX_PER_Transaction(uint256) (First100Pool.sol#1096-1099) is not in mixedCase
Variable AngrySheepClub.MAX_PER_Transaction (First100Pool.sol#922) is not in mixedCase
Variable AngrySheepClub.PRICE (First100Pool.sol#923) is not in mixedCase
Constant AngrySheepClub.TotalCollectionSize_ (First100Pool.sol#925) is not in UPPER_CASE_WITH_UNDERSCORES
Constant AngrySheepClub.PresaleSize (First100Pool.sol#926) is not in UPPER_CASE_WITH_UNDERSCORES
Variable AngrySheepClub.SubPresaleAmount (First100Pool.sol#927) is not in mixedCase
Variable AngrySheepClub.MintLimit (First100Pool.sol#928) is not in mixedCase
Function First100Pool.claim_First100_RewardM(uint256,bytes32[]) (First100Pool.sol#1124-1136) is not in mixedCase
Parameter First100Pool.claim_First100_RewardM(uint256,bytes32[])._tokenId (First100Pool.sol#1124) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Reentrancy in First100Pool.claim_First100_RewardM(uint256,bytes32[]) (First100Pool.sol#1124-1136):
        External calls:
        - address(msg.sender).transfer(_userCut) (First100Pool.sol#1133)
        State variables written after the call(s):
        - claimuserStatus[msg.sender][newTokenId] = true (First100Pool.sol#1134)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

First100Pool.ascContract (First100Pool.sol#1111) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
First100Pool.sol analyzed (18 contracts with 84 detectors), 56 result(s) found
```

Slither log >> HonoraryMembersPool.sol

```
HonoraryMembersPool.numberMinted(address).owner (HonoraryMembersPool.sol#992) shadows:
        - Ownable.owner() (HonoraryMembersPool.sol#357-359) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Variable 'ERC721A._checkOnERC721Received(address,address,uint256,bytes).retval (HonoraryMembersPool.sol#826)' in ERC721A._chec
kOnERC721Received(address,address,uint256,bytes) (HonoraryMembersPool.sol#817-840) potentially used before declaration: retval
 == IERC721Receiver(to).onERC721Received.selector (HonoraryMembersPool.sol#827)
Variable 'ERC721A._checkOnERC721Received(address,address,uint256,bytes).reason (HonoraryMembersPool.sol#828)' in ERC721A._chec
kOnERC721Received(address,address,uint256,bytes) (HonoraryMembersPool.sol#817-840) potentially used before declaration: reason
.length == 0 (HonoraryMembersPool.sol#829)
Variable 'ERC721A._checkOnERC721Received(address,address,uint256,bytes).reason (HonoraryMembersPool.sol#828)' in ERC721A._chec
kOnERC721Received(address,address,uint256,bytes) (HonoraryMembersPool.sol#817-840) potentially used before declaration: revert
(uint256,uint256)(32 + reason,mload(uint256)(reason)) (HonoraryMembersPool.sol#833)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
```

```
Address.isContract(address) (HonoraryMembersPool.sol#242-248) uses assembly
        - INLINE ASM (HonoraryMembersPool.sol#244-246)
Address.verifyCallResult(bool,bytes,string) (HonoraryMembersPool.sol#317-335) uses assembly
        - INLINE ASM (HonoraryMembersPool.sol#327-330)
ERC721A._checkOnERC721Received(address,address,uint256,bytes) (HonoraryMembersPool.sol#817-840) uses assembly
        - INLINE ASM (HonoraryMembersPool.sol#832-834)
MerkleProof._efficientHash(bytes32,bytes32) (HonoraryMembersPool.sol#876-882) uses assembly
        - INLINE ASM (HonoraryMembersPool.sol#877-881)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

HonoraryMembersPool.deposit() (HonoraryMembersPool.sol#955-957) compares to a boolean constant:
        -require(bool,string)(iscontractDepositPaused == false,deposit is paused) (HonoraryMembersPool.sol#956)
HonoraryMembersPool.claim_HonoraryMembers_Reward(uint256) (HonoraryMembersPool.sol#960-968) compares to a boolean constant:
        -require(bool,string)(claimuserStatus[msg.sender] == false,already claimed) (HonoraryMembersPool.sol#962)
HonoraryMembersPool.claim_HonoraryMembers_Reward(uint256) (HonoraryMembersPool.sol#960-968) compares to a boolean constant:
        -require(bool,string)(isclaimRewardPaused == false,claimReward is paused) (HonoraryMembersPool.sol#961)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```

```
Low level call in Address.sendValue(address,uint256) (HonoraryMembersPool.sol#249-254):
        - (success) = recipient.call{value: amount}() (HonoraryMembersPool.sol#252)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (HonoraryMembersPool.sol#275-286):
        - (success,returndata) = target.call{value: value}(data) (HonoraryMembersPool.sol#284)
Low level call in Address.functionStaticCall(address,bytes,string) (HonoraryMembersPool.sol#291-300):
        - (success,returndata) = target.staticcall(data) (HonoraryMembersPool.sol#298)
Low level call in Address.functionDelegateCall(address,bytes,string) (HonoraryMembersPool.sol#306-315):
        - (success,returndata) = target.delegatecall(data) (HonoraryMembersPool.sol#313)
Low level call in HonoraryMembersPool.withdrawMoney() (HonoraryMembersPool.sol#1003-1008):
        - (nfc) = address(0xa91EbE0e365B5cdfC5599a60d41D7bF8d93aEe06).call{value: address(this).balance * 20 / 100}() (Honorar
yMembersPool.sol#1004)
        - (artist) = address(0x2b838Ed253f393466D1330e34Fcb78e4F77d12a8).call{value: address(this).balance * 20 / 100}() (Hono
raryMembersPool.sol#1006)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter ERC721A.safeTransferFrom(address,address,uint256,bytes)._data (HonoraryMembersPool.sol#699) is not in mixedCase
Function HonoraryMembersPool.claim_HonoraryMembers_Reward(uint256) (HonoraryMembersPool.sol#960-968) is not in mixedCase
Parameter HonoraryMembersPool.claim_HonoraryMembers_Reward(uint256)._tokenId (HonoraryMembersPool.sol#960) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Reentrancy in HonoraryMembersPool.claim_HonoraryMembers_Reward(uint256) (HonoraryMembersPool.sol#960-968):
        External calls:
        - address(msg.sender).transfer(_userCut) (HonoraryMembersPool.sol#966)
        State variables written after the call(s):
        - claimuserStatus[msg.sender] = true (HonoraryMembersPool.sol#967)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

HonoraryMembersPool.merkleRoot (HonoraryMembersPool.sol#909) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
HonoraryMembersPool.sol analyzed (17 contracts with 84 detectors), 48 result(s) found
```

Slither log >> OGMintersPool.sol

```
AngrySheepClub.numberMinted(address).owner (OGMintersPool.sol#1066) shadows:
        - Ownable.owner() (OGMintersPool.sol#407-409) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

AngrySheepClub.setSubPresaleAmount(uint256) (OGMintersPool.sol#1052-1054) should emit an event for:
        - SubPresaleAmount = s (OGMintersPool.sol#1053)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Variable 'ERC721A._checkOnERC721Received(address,address,uint256,bytes).retval (OGMintersPool.sol#828)' in ERC721A._checkOnERC
721Received(address,address,uint256,bytes) (OGMintersPool.sol#819-842) potentially used before declaration: retval == IERC721R
eceiver(to).onERC721Received.selector (OGMintersPool.sol#829)
Variable 'ERC721A._checkOnERC721Received(address,address,uint256,bytes).reason (OGMintersPool.sol#830)' in ERC721A._checkOnERC
721Received(address,address,uint256,bytes) (OGMintersPool.sol#819-842) potentially used before declaration: reason.length == 0
 (OGMintersPool.sol#831)
Variable 'ERC721A._checkOnERC721Received(address,address,uint256,bytes).reason (OGMintersPool.sol#830)' in ERC721A._checkOnERC
721Received(address,address,uint256,bytes) (OGMintersPool.sol#819-842) potentially used before declaration: revert(uint256,uin
t256)(32 + reason,mload(uint256)(reason)) (OGMintersPool.sol#835)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
```

```
Low level call in Address.functionDelegateCall(address,bytes,string) (OGMintersPool.sol#356-365):
        - (success,returndata) = target.delegatecall(data) (OGMintersPool.sol#363)
Low level call in AngrySheepClub.withdrawMoney() (OGMintersPool.sol#1077-1080):
        - (NFC) = msg.sender.call{value: address(this).balance}() (OGMintersPool.sol#1078)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter ERC721A.safeTransferFrom(address,address,uint256,bytes)._data (OGMintersPool.sol#701) is not in mixedCase
Function AngrySheepClub.PresaleMint(uint256,bytes32[]) (OGMintersPool.sol#1006-1028) is not in mixedCase
Parameter AngrySheepClub.changeMintPrice(uint256)._newPrice (OGMintersPool.sol#1082) is not in mixedCase
Function AngrySheepClub.changeMAX_PER_Transaction(uint256) (OGMintersPool.sol#1086-1089) is not in mixedCase
Variable AngrySheepClub.MAX_PER_Transaction (OGMintersPool.sol#912) is not in mixedCase
Variable AngrySheepClub.PRICE (OGMintersPool.sol#913) is not in mixedCase
Constant AngrySheepClub.TotalCollectionSize_ (OGMintersPool.sol#915) is not in UPPER_CASE_WITH_UNDERSCORES
Constant AngrySheepClub.PresaleSize (OGMintersPool.sol#916) is not in UPPER_CASE_WITH_UNDERSCORES
Variable AngrySheepClub.SubPresaleAmount (OGMintersPool.sol#917) is not in mixedCase
Variable AngrySheepClub.MintLimit (OGMintersPool.sol#918) is not in mixedCase
Function OGMintersPool.claim_OGMinters_RewardM(uint256,bytes32[]) (OGMintersPool.sol#1116-1128) is not in mixedCase
Parameter OGMintersPool.claim_OGMinters_RewardM(uint256,bytes32[])._tokenId (OGMintersPool.sol#1116) is not in mixedCase
Function OGMintersPool.claim_OGMinters_Reward(uint256) (OGMintersPool.sol#1130-1142) is not in mixedCase
Parameter OGMintersPool.claim_OGMinters_Reward(uint256)._tokenId (OGMintersPool.sol#1130) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Reentrancy in OGMintersPool.claim_OGMinters_Reward(uint256) (OGMintersPool.sol#1130-1142):
        External calls:
        - address(msg.sender).transfer(_userCut) (OGMintersPool.sol#1139)
        State variables written after the call(s):
        - claimuserStatus[msg.sender][newTokenId] = true (OGMintersPool.sol#1140)
Reentrancy in OGMintersPool.claim_OGMinters_RewardM(uint256,bytes32[]) (OGMintersPool.sol#1116-1128):
        External calls:
        - address(msg.sender).transfer(_userCut) (OGMintersPool.sol#1125)
        State variables written after the call(s):
        - claimuserStatus[msg.sender][newTokenId] = true (OGMintersPool.sol#1126)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

OGMintersPool.ascContract (OGMintersPool.sol#1105) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
OGMintersPool.sol analyzed (18 contracts with 84 detectors), 61 result(s) found
```

Slither log >> AngrySheepDistribution.sol

```
AngrySheepClub.numberMinted(address).owner (AngrySheepDistribution.sol#1073) shadows:
        - Ownable.owner() (AngrySheepDistribution.sol#361-363) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

AngrySheepClub.setSubPresaleAmount(uint256) (AngrySheepDistribution.sol#1059-1061) should emit an event for:
        - SubPresaleAmount = s (AngrySheepDistribution.sol#1060)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

AngrySheepDistribution.setOGMinterspoolAddress(address)._OGMinterspoolAddress (AngrySheepDistribution.sol#1245) lacks a zero-c
heck on :
                - OGMintersPoolAddress = _OGMinterspoolAddress (AngrySheepDistribution.sol#1246)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Variable 'ERC721A._checkOnERC721Received(address,address,uint256,bytes).retval (AngrySheepDistribution.sol#863)' in ERC721A._c
heckOnERC721Received(address,address,uint256,bytes) (AngrySheepDistribution.sol#854-877) potentially used before declaration:
retval == IERC721Receiver(to).onERC721Received.selector (AngrySheepDistribution.sol#864)
Variable 'ERC721A._checkOnERC721Received(address,address,uint256,bytes).reason (AngrySheepDistribution.sol#865)' in ERC721A._c
heckOnERC721Received(address,address,uint256,bytes) (AngrySheepDistribution.sol#854-877) potentially used before declaration:
reason.length == 0 (AngrySheepDistribution.sol#866)
Variable 'ERC721A._checkOnERC721Received(address,address,uint256,bytes).reason (AngrySheepDistribution.sol#865)' in ERC721A._c
heckOnERC721Received(address,address,uint256,bytes) (AngrySheepDistribution.sol#854-877) potentially used before declaration:
revert(uint256,uint256)(32 + reason,mload(uint256)(reason)) (AngrySheepDistribution.sol#870)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
```

```
AngrySheepDistribution.ogmintersContract (AngrySheepDistribution.sol#1177) is set pre-construction with a non-constant functio
n or state variable:
        - OGMintersPool(OGMintersPoolAddress)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state

Pragma version^0.8.4 (AngrySheepDistribution.sol#9) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (AngrySheepDistribution.sol#251-256):
        - (success) = recipient.call{value: amount}() (AngrySheepDistribution.sol#254)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (AngrySheepDistribution.sol#277-288):
        - (success,returndata) = target.call{value: value}(data) (AngrySheepDistribution.sol#286)
Low level call in Address.functionStaticCall(address,bytes,string) (AngrySheepDistribution.sol#293-302):
        - (success,returndata) = target.staticcall(data) (AngrySheepDistribution.sol#300)
Low level call in Address.functionDelegateCall(address,bytes,string) (AngrySheepDistribution.sol#308-317):
        - (success,returndata) = target.delegatecall(data) (AngrySheepDistribution.sol#315)
Low level call in AngrySheepClub.withdrawMoney() (AngrySheepDistribution.sol#1084-1087):
        - (NFC) = msg.sender.call{value: address(this).balance}() (AngrySheepDistribution.sol#1085)
Low level call in AngrySheepDistribution.sendToWallet(address,uint256) (AngrySheepDistribution.sol#1238-1242):
        - (success) = recipient.call{value: amount}() (AngrySheepDistribution.sol#1240)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Reentrancy in AngrySheepDistribution.claimHolderMemberReward(uint256,bytes32[]) (AngrySheepDistribution.sol#1208-1221):
        External calls:
        - address(msg.sender).transfer(_userCut) (AngrySheepDistribution.sol#1219)
        State variables written after the call(s):
        - claimuserStatus[msg.sender][newTokenId] = true (AngrySheepDistribution.sol#1220)
Reentrancy in AngrySheepDistribution.claimHolderMemberRewardM(uint256,bytes32[]) (AngrySheepDistribution.sol#1192-1205):
        External calls:
        - address(msg.sender).transfer(_userCut) (AngrySheepDistribution.sol#1203)
        State variables written after the call(s):
        - claimuserStatus[msg.sender][newTokenId] = true (AngrySheepDistribution.sol#1204)
Reentrancy in OGMintersPool.claim_OGMinters_Reward(uint256) (AngrySheepDistribution.sol#1135-1147):
        External calls:
        - address(msg.sender).transfer(_userCut) (AngrySheepDistribution.sol#1144)
        State variables written after the call(s):
        - claimuserStatus[msg.sender][newTokenId] = true (AngrySheepDistribution.sol#1145)
Reentrancy in OGMintersPool.claim_OGMinters_RewardM(uint256,bytes32[]) (AngrySheepDistribution.sol#1121-1133):
        External calls:
        - address(msg.sender).transfer(_userCut) (AngrySheepDistribution.sol#1130)
        State variables written after the call(s):
        - claimuserStatus[msg.sender][newTokenId] = true (AngrySheepDistribution.sol#1131)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

AngrySheepDistribution.NFCAddress (AngrySheepDistribution.sol#1170) should be constant
AngrySheepDistribution.artistAddress (AngrySheepDistribution.sol#1169) should be constant
AngrySheepDistribution.ascContract (AngrySheepDistribution.sol#1176) should be constant
OGMintersPool.ascContract (AngrySheepDistribution.sol#1110) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

AngrySheepDistribution.ogmintersContract (AngrySheepDistribution.sol#1177) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
AngrySheepDistribution.sol analyzed (19 contracts with 84 detectors), 78 result(s) found
```

Slither log >> AngrySheepClub.sol

```
AngrySheepClub.numberMinted(address).owner (AngrySheepClub.sol#1071) shadows:
        - Ownable.owner() (AngrySheepClub.sol#360-362) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

AngrySheepClub.setSubPresaleAmount(uint256) (AngrySheepClub.sol#1057-1059) should emit an event for:
        - SubPresaleAmount = s (AngrySheepClub.sol#1058)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Variable 'ERC721A._checkOnERC721Received(address,address,uint256,bytes).retval (AngrySheepClub.sol#862)' in ERC721A._checkOnER
C721Received(address,address,uint256,bytes) (AngrySheepClub.sol#853-876) potentially used before declaration: retval == IERC72
1Receiver(to).onERC721Received.selector (AngrySheepClub.sol#863)
Variable 'ERC721A._checkOnERC721Received(address,address,uint256,bytes).reason (AngrySheepClub.sol#864)' in ERC721A._checkOnER
C721Received(address,address,uint256,bytes) (AngrySheepClub.sol#853-876) potentially used before declaration: reason.length ==
 0 (AngrySheepClub.sol#865)
Variable 'ERC721A._checkOnERC721Received(address,address,uint256,bytes).reason (AngrySheepClub.sol#864)' in ERC721A._checkOnER
C721Received(address,address,uint256,bytes) (AngrySheepClub.sol#853-876) potentially used before declaration: revert(uint256,u
int256)(32 + reason,mload(uint256)(reason)) (AngrySheepClub.sol#869)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
```

```
Reentrancy in ERC721A._safeMint(address,uint256,bytes) (AngrySheepClub.sol#750-782):
        External calls:
        - require(bool,string)(_checkOnERC721Received(address(0),to,updatedIndex,_data),ERC721A: transfer to non ERC721Receive
r implementer) (AngrySheepClub.sol#773-776)
                - IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (AngrySheepClub.sol#860-872)
        Event emitted after the call(s):
        - Transfer(address(0),to,updatedIndex) (AngrySheepClub.sol#772)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

ERC721A._transfer(address,address,uint256) (AngrySheepClub.sol#783-823) uses timestamp for comparisons
        Dangerous comparisons:
        - _ownerships[nextTokenId].addr == address(0) (AngrySheepClub.sol#812)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.isContract(address) (AngrySheepClub.sol#244-250) uses assembly
        - INLINE ASM (AngrySheepClub.sol#246-248)
Address.verifyCallResult(bool,bytes,string) (AngrySheepClub.sol#319-337) uses assembly
        - INLINE ASM (AngrySheepClub.sol#329-332)
MerkleProof._efficientHash(bytes32,bytes32) (AngrySheepClub.sol#406-412) uses assembly
        - INLINE ASM (AngrySheepClub.sol#407-411)
ERC721A._checkOnERC721Received(address,address,uint256,bytes) (AngrySheepClub.sol#853-876) uses assembly
        - INLINE ASM (AngrySheepClub.sol#868-870)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
Pragma version^0.8.4 (AngrySheepClub.sol#8) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (AngrySheepClub.sol#251-256):
        - (success) = recipient.call{value: amount}() (AngrySheepClub.sol#254)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (AngrySheepClub.sol#277-288):
        - (success,returndata) = target.call{value: value}(data) (AngrySheepClub.sol#286)
Low level call in Address.functionStaticCall(address,bytes,string) (AngrySheepClub.sol#293-302):
        - (success,returndata) = target.staticcall(data) (AngrySheepClub.sol#300)
Low level call in Address.functionDelegateCall(address,bytes,string) (AngrySheepClub.sol#308-317):
        - (success,returndata) = target.delegatecall(data) (AngrySheepClub.sol#315)
Low level call in AngrySheepClub.withdrawMoney() (AngrySheepClub.sol#1082-1085):
        - (NFC) = msg.sender.call{value: address(this).balance}() (AngrySheepClub.sol#1083)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter ERC721A.safeTransferFrom(address,address,uint256,bytes)._data (AngrySheepClub.sol#735) is not in mixedCase
Function AngrySheepClub.PresaleMint(uint256,bytes32[]) (AngrySheepClub.sol#1011-1033) is not in mixedCase
Parameter AngrySheepClub.changeMintPrice(uint256)._newPrice (AngrySheepClub.sol#1087) is not in mixedCase
Function AngrySheepClub.changeMAX_PER_Transaction(uint256) (AngrySheepClub.sol#1091-1094) is not in mixedCase
Variable AngrySheepClub.MAX_PER_Transaction (AngrySheepClub.sol#917) is not in mixedCase
Variable AngrySheepClub.PRICE (AngrySheepClub.sol#918) is not in mixedCase
Constant AngrySheepClub.TotalCollectionSize_ (AngrySheepClub.sol#920) is not in UPPER_CASE_WITH_UNDERSCORES
Constant AngrySheepClub.PresaleSize (AngrySheepClub.sol#921) is not in UPPER_CASE_WITH_UNDERSCORES
Variable AngrySheepClub.SubPresaleAmount (AngrySheepClub.sol#922) is not in mixedCase
Variable AngrySheepClub.MintLimit (AngrySheepClub.sol#923) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
AngrySheepClub.sol analyzed (17 contracts with 84 detectors), 50 result(s) found
```

Solidity Static Analysis

First100Pool.sol

Security

Transaction origin:

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

more

Pos: 1087:12:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in First100Pool.claim_First100_RewardM(uint256,bytes32[]): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1170:4:

Gas & Economy

Gas costs:

Gas requirement of function ERC721A.tokenByIndex 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: 571:2:

Email: audit@EtherAuthority.io

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: 904:8:

Miscellaneous

Constant/View/Pure functions:

AngrySheepClub.safeTransferFrom(address,address,uint256,bytes) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1001:4:

Similar variable names:

MerkleProof.verify(bytes32[],bytes32,bytes32) : Variables have very similar names "proof" and "root". Note: Modifiers are currently not considered by this static analysis.
Pos: 900:44:

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: 1177:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 1178:25:

HonoraryMembersPool.sol

Security

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases. If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.
more
Pos: 1016:18:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in HonoraryMembersPool.claim_HonoraryMembers_Reward(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1006:4:

Gas & Economy

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 904:8:

Miscellaneous

Constant/View/Pure functions:

HonoraryMembersPool.safeTransferFrom(address,address,uint256,bytes) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 985:4:

Similar variable names:

MerkleProof.verify(bytes32[],bytes32,bytes32) : Variables have very similar names "proof" and "root". Note: Modifiers are currently not considered by this static analysis.
Pos: 900:44:

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: 1053:4:

OGMintersPool.sol

Security

Transaction origin:

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

more

Pos: 1087:12:

Check-effects-interaction:

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

more

Pos: 1183:7:

Low level calls:

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

more

Pos: 1125:19:

Gas & Economy

Gas costs:

Gas requirement of function AngrySheepClub.OPERATOR_FILTER_REGISTRY 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: 178:4:

Gas costs:

Gas requirement of function OGMintersPool.claim_OGMinters_Reward 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: 1183:7:

Miscellaneous

Constant/View/Pure functions:

AngrySheepClub.safeTransferFrom(address,address,uint256,bytes) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1001:4:

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: 946:8:

AngrySheepDistribution.sol

Security

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases. If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.
more
Pos: 1083:12:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.
more
Pos: 1281:23:

Gas & Economy

Gas costs:

Gas requirement of function AngrySheepClub.withdrawMoney 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: 1120:2:

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: 420:8:

Miscellaneous

Constant/View/Pure functions:

AngrySheepClub.safeTransferFrom(address,address,uint256,bytes) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 997:4:

Data truncated:

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

Pos: 1280:21:

AngrySheepClub.sol

Security

Transaction origin:

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

more

Pos: 1082:12:

Low level calls:

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

more

Pos: 1120:19:

Gas & Economy

Gas costs:

Gas requirement of function AngrySheepClub.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: 1024:2:

Miscellaneous

Constant/View/Pure functions:

AngrySheepClub.safeTransferFrom(address,address,uint256,bytes) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 996:4:

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: 996:10:

Solhint Linter

First100Pool.sol

```
First100Pool.sol:176:28: Error: Parse error: mismatched input '('
expecting {';', '='}
First100Pool.sol:176:45: Error: Parse error: extraneous input ')'
expecting {';', '='}
First100Pool.sol:229:41: Error: Parse error: mismatched input '('
expecting {';', '='}
```

HonoraryMembersPool.sol

```
HonoraryMembersPool.sol:176:28: Error: Parse error: mismatched input
'(' expecting {';', '='}
HonoraryMembersPool.sol:176:45: Error: Parse error: extraneous input
')' expecting {';', '='}
HonoraryMembersPool.sol:229:41: Error: Parse error: mismatched input
'(' expecting {';', '='}
```

OGMintersPool.sol

```
OGMintersPool.sol:176:28: Error: Parse error: mismatched input '('
expecting {';', '='}
OGMintersPool.sol:176:45: Error: Parse error: extraneous input ')'
expecting {';', '='}
OGMintersPool.sol:229:41: Error: Parse error: mismatched input '('
expecting {';', '='}
```

AngrySheepDistribution.sol

```
AngrySheepDistribution.sol:176:28: Error: Parse error: mismatched
input '(' expecting {';', '='}
AngrySheepDistribution.sol:176:45: Error: Parse error: extraneous
input ')' expecting {';', '='}
AngrySheepDistribution.sol:229:41: Error: Parse error: mismatched
input '(' expecting {';', '='}
```

AngrySheepClub.sol

```
AngrySheepClub.sol:175:28: Error: Parse error: mismatched input '('
expecting {';', '='}
AngrySheepClub.sol:175:45: Error: Parse error: extraneous input ')'
expecting {';', '='}
AngrySheepClub.sol:228:41: Error: Parse error: mismatched input '('
expecting {';', '='}
```

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