Alethea AI Token Version 3.0 - Bonding Curves

Smart Contract Pre-Audit Check and Code Review

Version 1

Smart Contract Pre-Audit Check and Code Review

Prepared by: Darren Jensen

Version: 1.0
Date: 13th December 2023

Introduction

This document outlines the findings for smart contract code review for contracts in ai-protocol-contracts repo at commit SHA efef8eec and specifically focuses on the contracts in the `contracts/bonding_curves` folder and the `OpAliERC20v2` token contract. All associated test files and deployment scripts were also reviewed as part of the scope of work.

Impact

- **High** - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- **Medium** - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- **Low** - can lead to any kind of unexpected behaviour with some of the protocol's functionalities that's not so critical.

Likelihood

- **High** - attack path is possible with reasonable assumptions that mimic on-chain conditions and the cost of the attack is relatively low to the amount of funds that can be stolen or lost.
- **Medium** - only conditionally incentivized attack vector, but still relatively likely.
- **Low** - has too many or too unlikely assumptions or requires a huge stake by the attacker with little or no incentive.

Severity classification

Severity	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: High	High	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Low	Low	Low

Actions required by severity level

- **High** - client **must** fix the issue.
- **Medium** - client **should** fix the issue.
- **Low** - client **could** fix the issue.

- **Informational** - client **could** consider design/UX related decision
- **Recommendation** - client **could** have an internal team discussion on whether the recommendations provide any UX or security enhancement and if it is technically and economically feasible to implement the recommendations
- **Gas Findings** - client **could** consider implementing suggestions for better UX

Overview

Project Name	Alethea AI BondingCurves, TradeableShares & OpAliERC20v2
Repository	ai-protocol-contracts
Commit SHA	efef8eec
Documentation	Provided
Methods	Manual review & CLI review (Mythril, Slither, Solhint)

Contracts in Scope

In this report I have focused on all contracts in the contracts/bonding_curves directory plus the token/OpAliERC20v2.sol contract as follows:

AbstractShares.sol
FriendTechBondingCurve.sol
ProtocolFeeDistributorV1.sol
SharesFactoryV1.sol
BondingCurve.sol
SharesSubjectLib.sol
ERC20Shares.sol
HoldersRewardsDistributor.sol
RewardSystem.sol
TradeableShares.sol
ETHShares.sol
HoldersRewardsDistributorV1.sol
SharesFactory.sol
TypedStructLib.sol
token/OpAliERC20v2.sol

Issues found

Severity	Count
High risk	0
Medium risk	2
Low risk	4
Informational	17
Recommendations	0
Gas Findings	3

Medium Findings

[M-1] Potential to front run in `HoldersRewardsDistributorV1` affecting `__accept` function

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/contracts/bonding_curves/HoldersRewardsDistributorV1.sol#L168

Description

It's possible for a malicious user to affect the `accRewardPerShare` calculation by calling the `receive()` function in the same block before any buy / sell tx is mined. This would set the `lastRewardBlock` to the current block and the `feeAmount` sent into the contract would not be added to the `accRewardPerShare`.

Recommended Mitigation Steps

Consider protecting this function from front running. However, it's unlikely that an attacker would benefit from such an action other than purposely reducing the rewards for holders.

[M-2] If `owner` (via the Factory `sharesOwnerAddress`) is not set in the shares contract then it's not possible to manage

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/contracts/bonding_curves/SharesFactoryV1.sol#L619

Description

In the `__initSharesContract` function of the `SharesFactoryV1` contract the shares contracts are deployed using the `sharesOwnerAddress` as the owner. However, if the `sharesOwnerAddress` is a zero address then the deployed sharers contract will not be possible to manage.

Recommended Mitigation Steps

Ensure that the `sharesOwnerAddress` is set to a valid address to use as the owner of the shares contracts before calling the `__initSharesContract` function. Moreover, consider checking the deployment scripts to set the `sharesOwnerAddress` via a call to `setSharesOwnerAddress`.

Low Findings

[L-1] `ProtocolFeeDistributorV1` contract `updateRecipientsList` function does not check for duplicate recipient address

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/contracts/bonding_curves/ProtocolFeeDistributorV1.sol#L181

Description

It's possible to set duplicate recipients in `updateRecipientsList` function including setting all the recipients to the same address.

Recommended Mitigation Steps

Consider performing a uniqueness check in the input addresses of the `_recipients` array.

[L-2] Check the shares contract type before setting in `HoldersRewardsDistributorV1`

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/contracts/bonding_curves/HoldersRewardsDistributorV1.sol#L105

Description

Check the *shares contract* type before setting in the contract storage variable. If the `paymentToken` address is set in the `HoldersRewardsDistributorV1` contract then the *shares contract* should be `ERC20Shares` type, otherwise `ETHShares` type.

Recommended Mitigation Steps

Add a validation for sharesContractAddress points to a shares contract of the expected type (`ERC20Shares` or `ETHShares`).

[L-3] The `RewardSystem` proxy deployment sets the `rewardSystemType` incorrectly

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/deploy/v3_0/deploy-RewardSystem_Proxy.js#L48

Description

In the deployment script for the `RewardSystem` proxy the type is set to `TRUE` when it should be set to `FALSE` given that an `ERC20` token is set then it suggests this is an `ERC20` `rewardSystemType` deployment.

Recommended Mitigation Steps

Consider updating the deployment script so that the `rewardSystemType` is set correctly in the deployment.

[L-4] The `RewardSystem` proxy deployment is missing the ETH `rewardSystemType`.

Context

N/A

Description

There does not appear to be any deployment of the `RewardSystem` proxy for the `ETH rewardSystemType`.

Recommended Mitigation Steps

Consider adding a deployment script for the `ETH rewardSystemType`.

Informational Findings

[I-1] Validate the bridge address is trusted before updating role

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/contracts/token/OpAliERC20v2.sol#L46

Description

The role is applied to the bridge contract before validating the address is a known trusted bridge.

Recommended Mitigation Steps

Consider validating the bridge before assigning the role.

[I-2] Bonding curve price function differs slightly from the FriendTech version

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/contracts/bonding_curves/FriendTechBondingCurve.sol#L23

Description

Calculation for `sum2` is slightly different to the FriendTech version which sets `sum2` to 0 only when supply is 0 *and* amount is 1. In the AletheaAI implementation this has been changed to amount is <=1.

AletheaAI version

```
uint256 sum2 = s == 0 && a <= 1 ? 0 : (s + a - 1) * (s + a) * (2 * (s + a - 1) + 1) / 6;
```

FriendTech version

```
uint256 sum2 = s == 0 && a == 1 ? 0 : (s + a - 1) * (s + a) * (2 * (s + a - 1) + 1) / 6;
```

Recommended Mitigation Steps

Consider aligning the versions to be exactly the same.

[I-3] Missing detail in comment for keccak256 value

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/contracts/bonding_curves/TypedStructLib.sol#L17

Description

Missing comment showing the `keccak256` used to generate the hash.

Recommended Mitigation Steps

Consider adding comments as is for the `SharesSubject` type hash.

[I-4] Outdated comment in HoldersRewardsDistributor

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/contracts/bonding_curves/HoldersRewardsDistributor.sol#L40

Description

The comment regarding the fallback is outdated. The encoded data includes an `isBuy` bool and the amount is always positive (`uint256`).

Recommended Mitigation Steps

Consider updating the comment to reflect existing behaviour.

[I-5] Naming convention for contract interfaces

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/contracts/bonding_curves/HoldersRewardsDistributor.sol#L40

Description

It's not clear from the name of the contract this is an interface.

Recommended Mitigation Steps

Consider using 'I' prefix for *all* interface names. So, for example, renaming `HoldersRewardsDistributor` to `IHoldersRewardsDistributor`.

[I-6] TODO comments in contract code

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/contracts/bonding_curves/HoldersRewardsDistributor.sol#L122

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/deploy/v3_0/deploy-SharesFactory_Proxy.js#L16

Description

`TODO` comment in code suggests a potential change to the `claimTheReward` function.

Recommended Mitigation Steps

Consider removing the `TODO` comment.

[I-7] Include an .nvmrc file to set the node version to 16 for devs

Context

N/A

Description

Developers can benefit from an `.nvmrc` file at the root of the project folder so that can potentially trigger NVM to automatically switch to the version of Node JS as specified in the `.nvmrc` file.

Recommended Mitigation Steps

Consider adding an `.nvmrc` file to the repo with the contents set to the current version of the Node being used. The file can be generated like so and then added to the project git repo:

```
node -v > .nvmrc
```

[I-8] Describe block in test could be an it block

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/test/bonding_curves/factory.js#L365

Description

The `describe` block in this example test could be an `it` block.

Recommended Mitigation Steps

Consider changing to an `it` block.

[I-9] Consider using `console.warn`

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/test/bonding_curves/gas_usage_factory_shares_ERC20.js#L93

Description

In the gas optimization tests there is a `console.log` output for when the gas used is less than the expected amount. Using `console.warn` may be better to highlight this issue when it is included in the test logs.

Recommended Mitigation Steps

Consider changing `console.log` to `console.warn`.

[I-10] JS version of `get_price` in tests is slightly different to the `getPrice` implementation in `FriendTechBondingCurve`

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/test/bonding_curves/include/curves.js#L16

Description

This appears only when the given supply is > 1 and the amount is > 0 then the resulting price from these function calls always differs.

POC

```
describe("Checking impl of get price in JS vs Solidity", function() {
       let bc;
       beforeEach(async function() {
           const FriendTechBondingCurve = artifacts.require("FriendTechBondingCurve");
           bc = await FriendTechBondingCurve.new()
       });
       it.only("calling get_price in JS vs getPrice in Solidity", async function() {
           const supply = 2;
           const amount = 1;

           priceJs = await get_price(supply, amount);
           console.log("Price (JS): ", priceJs.toString());
           priceSol = await bc.getPrice(supply, amount);
           console.log("Price (SOL): ", priceSol.toString());
       });
   })
```

The result of the above POC test is:

```
Price (JS):  312500000000000
Price (SOL):  250000000000000
```

Recommended Mitigation Steps

Correct the JS implementation of `get_price` used in the tests to exactly match the implementation of the Solidity implementation of `getPrice`.

[I-11] Missing associated parent or Interface contract

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/contracts/bonding_curves/ProtocolFeeDistributorV1.sol#L17

Description

There is no corresponding parent or Interface contract as there are with the contracts (e.g. `HoldersRewardsDistributorV1` has the `HoldersRewardsDistributor` Interface)

Recommended Mitigation Steps

Consider adding a `IProtocolFeeDistributor` contract for the `ProtocolFeeDistributorV1`.

[I-12] `receive` function does not need to be marked `virtual`

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/contracts/bonding_curves/ProtocolFeeDistributorV1.sol#L107

Description

The `receive` function in the ProtocolFeeDistributorV1 contract does not need to be marked as `virtual`.

Recommended Mitigation Steps

Consider removing the `virtual` keyword.

[I-13] Emit event when `sharesContractAddress` is set in `HoldersRewardsDistributorV1`

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/contracts/bonding_curves/HoldersRewardsDistributorV1.sol#L108

Description

The `sharesContractAddress` is set in `HoldersRewardsDistributorV1` without any event being fired.

Recommended Mitigation Steps

Consider emitting an event when `sharesContractAddress` is set in `HoldersRewardsDistributorV1` contract.

[I-14] Possible to pass any nonce to `rewindNonce` function that is greater than the current which would leave gaps

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/contracts/bonding_curves/SharesFactoryV1.sol#L720

Description

In the `rewindNonce` function of the `SharesFactoryV1` contract it's possible to pass in any nonce that is greater than the current which would leave gaps.

Recommended Mitigation Steps

Consider restricting the new nonce to be exactly one greater than the current nonce.

[I-15] Comment mentions `ROLE_PROTOCOL_FEE_MANAGER` but the values set is for `ROLE_SUBJECT_FEE_MANAGER`

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/contracts/bonding_curves/SharesFactoryV1.sol#L145

Description

Comment refers to a different variable name..

Recommended Mitigation Steps

Consider correcting the comment to properly reflect the variabile name.

[I-16] Potentially missing a deployment dependency in `setup-SharesFactory` script.

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/deploy/v3_0/setup-SharesFactory.js#L253

Description

Perhaps `upgrade-ProtocolFeeDistributorV1` should also be listed as a dependency in this script.

Recommended Mitigation Steps

Consider adding `upgrade-ProtocolFeeDistributorV1` also be listed as a dependency in this script.

[I-17] `determineImplementationType` will always return `ImplementationType.ETH` if any other address is passed

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/contracts/bonding_curves/SharesFactoryV1.sol#L820

Description

If any address is passed into the function (except a valid `ERC20` shares address) then the function will always return `ImplementationType.ETH` result.

Recommended Mitigation Steps

Consider performing a check for a valid `TradeableShares` contract before defaulting to `ImplementationType.ETH`.

Gas Findings

[G-1] Can reuse `sharesSupply` value

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/contracts/bonding_curves/ETHShares.sol#L191

Description

The calculation for `sharesSupply` is performed twice.

Recommended Mitigation Steps

The `sharesSupply` can be passed directly to the `getPrice` function to save a little gas

[G-2] Function call can be avoided by inline code

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/contracts/bonding_curves/HoldersRewardsDistributorV1.sol#L183

Description

Calling `pendingReward` in the `claimTheReward` function can be avoided by moving this function code inline and therefore avoiding the function hop which will save some gas.

Recommended Mitigation Steps

Consider moving the `pendingReward` function logic inline of the `claimTheReward` function to save some gas when calling this function.

[G-3] Checking issuer address is not address(0) multiple times

Context

https://github.com/AletheaAI/ai-protocol-contracts/blob/efef8eecd87f18b88ef8ceee7dc68be73664f187/contracts/bonding_curves/SharesFactoryV1.sol#L549

Description

The assert statement (`assert(issuer != address(0));`) comes after a `require` statement that is run if the issuer is a zero address.

Recommended Mitigation Steps

Consider removing the `assert` statement on the line shown in the context link above.

Contract versions used in audit

This report was conducted by using the contracts in SHA efef8eec. Please note that the author ***did not make any modifications to the Smart Contracts***. All the SHA-256 smart contract file fingerprints are shown in Appendix A of this document and can be recalculated if needed to ensure the validity and expected code version of the contracts.

White Paper / Specifications Document

The auditor reviewed the TradableShares README as provided in the shared repo.

Test Run

The auditor built the contracts using hardhat and ran all the tests which are passing (`5110 passing (24m)`).

All the contracts were compiled and the test run executed successfully with **all tests passing**.

Test Coverage

Below shows the output of the test coverage report for the bonding curve contracts.

```
-----------------------------------------|----------|----------|----------|----------|----------------|
File                                     |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
-----------------------------------------|----------|----------|----------|----------|----------------|
 bonding_curves/                         |    98.77 |    91.84 |    98.25 |    99.05 |                |
  AbstractShares.sol                     |      100 |       90 |      100 |      100 |                |
  BondingCurve.sol                       |      100 |      100 |      100 |      100 |                |
  ERC20Shares.sol                        |      100 |    90.63 |      100 |      100 |                |
  ETHShares.sol                          |      100 |      100 |      100 |      100 |                |
  FriendTechBondingCurve.sol             |      100 |      100 |      100 |      100 |                |
  HoldersRewardsDistributor.sol          |      100 |      100 |      100 |      100 |                |
```

```
  HoldersRewardsDistributorV1.sol         |    97.62 |    86.84 |      100 |    98.36 |            260 |
  ProtocolFeeDistributorV1.sol            |    96.97 |     87.5 |     87.5 |     97.5 |            231 |
  RewardSystem.sol                        |      100 |    93.75 |       80 |      100 |                |
  SharesFactory.sol                       |      100 |      100 |      100 |      100 |                |
  SharesFactoryV1.sol                     |    98.97 |    94.12 |      100 |    99.15 |            668 |
  SharesSubjectLib.sol                    |    94.74 |    83.33 |      100 |    95.45 |             90 |
  TradeableShares.sol                     |      100 |      100 |      100 |      100 |                |
  TypedStructLib.sol                      |      100 |      100 |      100 |      100 |                |
------------------------------------------|----------|----------|----------|----------|----------------|
All files                                 |     98.1 |    92.61 |    97.63 |    98.07 |                |
------------------------------------------|----------|----------|----------|----------|----------------|
```

Linting

Linting is a valuable tool for finding potential issues in smart contract code. It can find stylistic errors, violations of programming conventions, and unsafe constructs in your code. There are many great linters available, such as Solhint. Linting can help find potential problems, even security problems such as re-entrancy vulnerabilities before they become costly mistakes.

After installation, Solhint can be run via the terminal as follows.

```
solhint 'contracts/bonding_curves/*.sol'
```

This command will run `solhint` for the contracts in the root of the project directory. I have run the above command and included the output in the Appendix section of this report. The only issue identified by `solhint` was the length of the line in some instances. The full report is shown in the Appendix section of this report.

I recommend running the `solhint` linter, either via a Git commit hook or as part of an integration with the developer IDE so that the recommendations can be checked on every code change. **NOTE: the above issues are low priority and would not have any impact if not modified.**

Slither Security Analysis

Auditor uses Slither (version 0.9.6) static analyzer tool. The slither cli was run against all contracts in scope of the project.

For each of the above contracts the correct version of `solc` using `solc-select` and then run the `slither` command to analyze specific contract under test:

```
solc-select use 0.8.15

slither --checklist --exclude-informational --exclude-low --solc-remaps
"@openzeppelin/=node_modules/@openzeppelin/" contracts/bonding_curves  2>&1 |
tee slither-bonding-curves.md
```

The final report is available in the Appendix here.

Mythril Security Analysis

The Mythril security analysis reports were run using the MythX service in `standard` mode using the following commands:

```
mythx analyze --mode standard --remap-import
"@openzeppelin/=$(pwd)/node_modules/@openzeppelin/" --solc-version
0.8.15 contracts/bonding_curves/*.sol
```

The analysis was performed via the MythX services and the pdf reports generated and downloaded.

NOTE: One issue is the `totalClaimedReward` variable in `RewardSystem` does not have its visibility set making the default visibility of this variable `internal`. Only other issues were around `floating pragma`.

Each contract has its own individual report (that can be downloaded as PDF) for each submitted contract to the MythX service. These reports are included in a separate ZIP file along with this main report.

Conclusion

All contracts reviewed in scope are well written and organized.

There are only two *medium* issues identified and a number of low and informational issues which should be easily addressed.

All tests are passing and there are only minor issues reported with the test code and setup. All deployment scripts under V3 were reviewed and mostly are good with some minor issues identified and listed earlier in this document.

It's worth considering further analysis of the contracts logic and state by performing some specific invariant tests against the contracts such as performing thousands of buy / sell trades and checking that certain invariants hold. This work could be carried out as part of a subsequent report of work by the auditor using invariant testing tools in Foundry.

Appendix A

LoC (Lines of Code)

The [cloc utility]was used to determine the lines of code under review. The utility excludes empty lines and comments to leave a count of auditable lines of code in each contract. Since all contracts in scope are under the `contracts/bonding_curves` folder the `cloc` command was run once with the output as follows:

```
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Solidity                        14            408           2015           1234
-------------------------------------------------------------------------------
SUM:                            14            408           2015           1234
-------------------------------------------------------------------------------
```

SHA-256 File Fingerprints

To generate the SHA-256 fingerprint for *all the smart contracts* in a directory run:

```
shasum -a 256 contracts/bonding_curves/*.sol
```

Which outputs the following:

```
f1a6e2f1ebefe046a8d36e156075d7bd6044ab35351616f06e30889de9d4ed67  AbstractShares.sol
714f43a2392767fd7ce1b73fbbbe8239c8be7b0ae5f94dd7877fccccc7eb1704  BondingCurve.sol
9b8faaf1e4cf06a84b333a4522cc8b16c6260014249fc38b1f9ed27879c2dc9a  ERC20Shares.sol
71e05fc644e2c694ded1cec1e593ec1494a71aaed69422c73923e9eadd878ccd  ETHShares.sol
41703d3a679113cd7288f8183458910a673751205ad7152333db824844164690  FriendTechBondingCurve.sol
5eb93e8eb8a876c99fd173e58200d326a7f2c468963ce67a2b633e2ce786149a
HoldersRewardsDistributor.sol
d64e3589223db5cd9ec30531279811fc24e7bcc189b0e8731d8a46ba38609cce
HoldersRewardsDistributorV1.sol
25a8de34654b050dda5d5cc88532c1c1adcf8e58bef758d180c192884c02b867  ProtocolFeeDistributorV1.sol
b1d3b603a6c1ce9d278fd07377efd81d51c7a94d4251a59bc5bb423ce9bd2ca4  RewardSystem.sol
5c6ad032d4d723deb8fead7668b2bb5fe1b819dd658fa0fd4bad16260fe583fd  SharesFactory.sol
6db909c21add8f3d2233fd55b145ca37411f2c1acb526d4a28783e3e6ccd4e09  SharesFactoryV1.sol
a38a79130042732606c395ba9432d13606c395cec64ba7fc11db36c00bc28eb5  SharesSubjectLib.sol
f3e5ee87e5916ec43aab6516f704993d43f722d7ec2d4f327ec1f2ef2a44efd5  TradeableShares.sol
5bc5cc8ba35c704763894c43b79142f6e7702b1dc39fbe7d2d5e6700d68a7ed8  TypedStructLib.sol
```

Solhint Report

The command to run `solhint` for all tokens contracts is as follows:

```
solhint 'contracts/bonding_curves/*.sol'
```

The output of the command is as follows. Note there are no issues reported other than a max line length violation.

```
contracts/bonding_curves/ERC20Shares.sol
  265:2  error  Line length must be no more than 120 but current length is 128
max-line-length

contracts/bonding_curves/SharesFactoryV1.sol
  417:2  error  Line length must be no more than 120 but current length is 124
max-line-length
  637:2  error  Line length must be no more than 120 but current length is 127
max-line-length

✖ 3 problems (3 errors, 0 warnings)
```

Test Run Report

Tests were run for the contracts under scope using the testing tools provided in the repo (hardhat). The command to run the tests are as follows:

```
npx hardhat test test/bonding_curves/*.js
```

Mythril Report

Below is the console output from the Mythril Report. Each report is available as a PDF file which will be included in a separate ZIP file along with this report. The output below is for a quick summary only. Below is a summary of the reports:

```
Report for contracts/bonding_curves/AbstractShares.sol
https://dashboard.mythx.io/#/console/analyses/36580b47-70f7-4976-b470-82c81a8fb3aa
```

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

```
Report for contracts/bonding_curves/ERC20Shares.sol
https://dashboard.mythx.io/#/console/analyses/df8d187f-0e80-44e3-9f5d-c9dba4b1ce65
```

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

```
Report for contracts/bonding_curves/ETHShares.sol
https://dashboard.mythx.io/#/console/analyses/e8314a0a-64bb-4543-aa93-2eb2dee700a8
```

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
171	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
202	(SWC-107) Reentrancy	Low	Read of persistent state following external call.

```
Report for FriendTechBondingCurve.sol
https://dashboard.mythx.io/#/console/analyses/cf3a7145-47e9-4e5d-88d8-02f12dd9ffc3
```

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Report for contracts/bonding_curves/HoldersRewardsDistributorV1.sol
https://dashboard.mythx.io/#/console/analyses/ecd1f887-cea9-4e47-ade7-dc8b8d8fb538

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Report for contracts/bonding_curves/ProtocolFeeDistributorV1.sol
https://dashboard.mythx.io/#/console/analyses/02ee81bd-93a3-4a67-bd88-baf6515eb160

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Report for contracts/bonding_curves/RewardSystem.sol
https://dashboard.mythx.io/#/console/analyses/a6bc935d-6186-411e-a6fd-39ab59ff76f9

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
42	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.

Report for contracts/bonding_curves/SharesFactoryV1.sol
https://dashboard.mythx.io/#/console/analyses/91fa950d-8f13-43b6-88a5-55220b97cc42

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Report for bonding_curves/SharesSubjectLib.sol
https://dashboard.mythx.io/#/console/analyses/26937ce0-7f87-43fc-89ba-ff05c32ef3fb

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Report for bonding_curves/TypedStructLib.sol
https://dashboard.mythx.io/#/console/analyses/7dd5ef55-0490-4006-8e0f-cb4997a43dc3

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

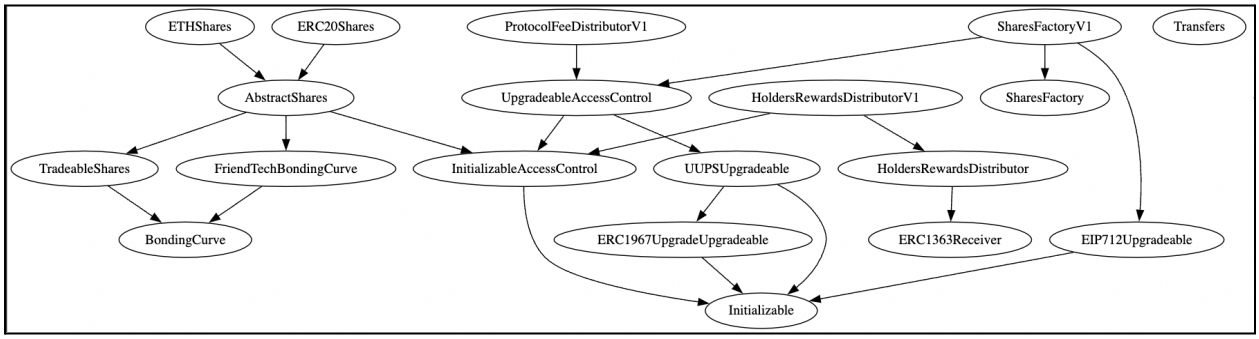
Slither Report

Below are the Slither Report summary output for all contracts under the scope of this audit. The full markdown files are shared separately. **NOTE** the reports were generated with `--exclude-informational` & `--exclude-low` flags set to exclude issues at these levels. Conclusion is that the report only shows false positives.

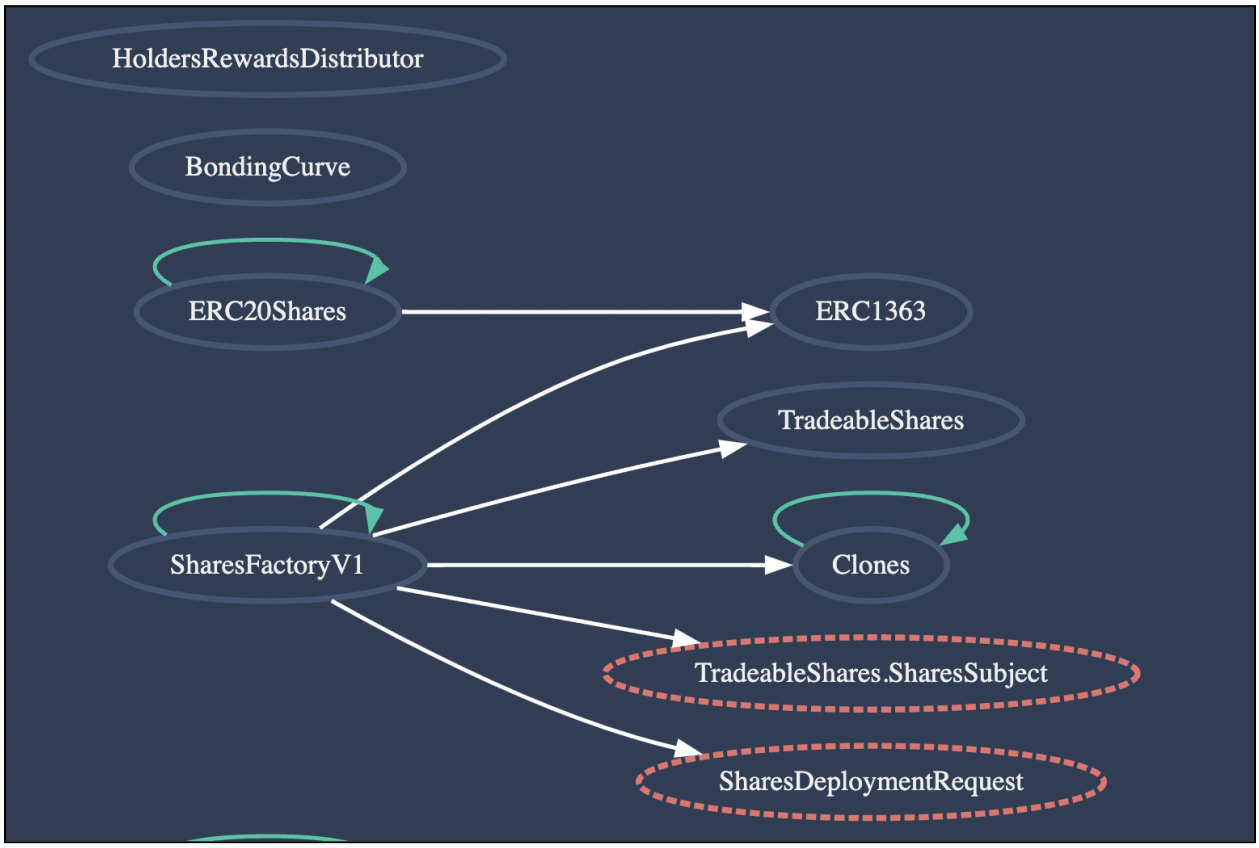
`controlled-delegatecall (3 results) (High)`

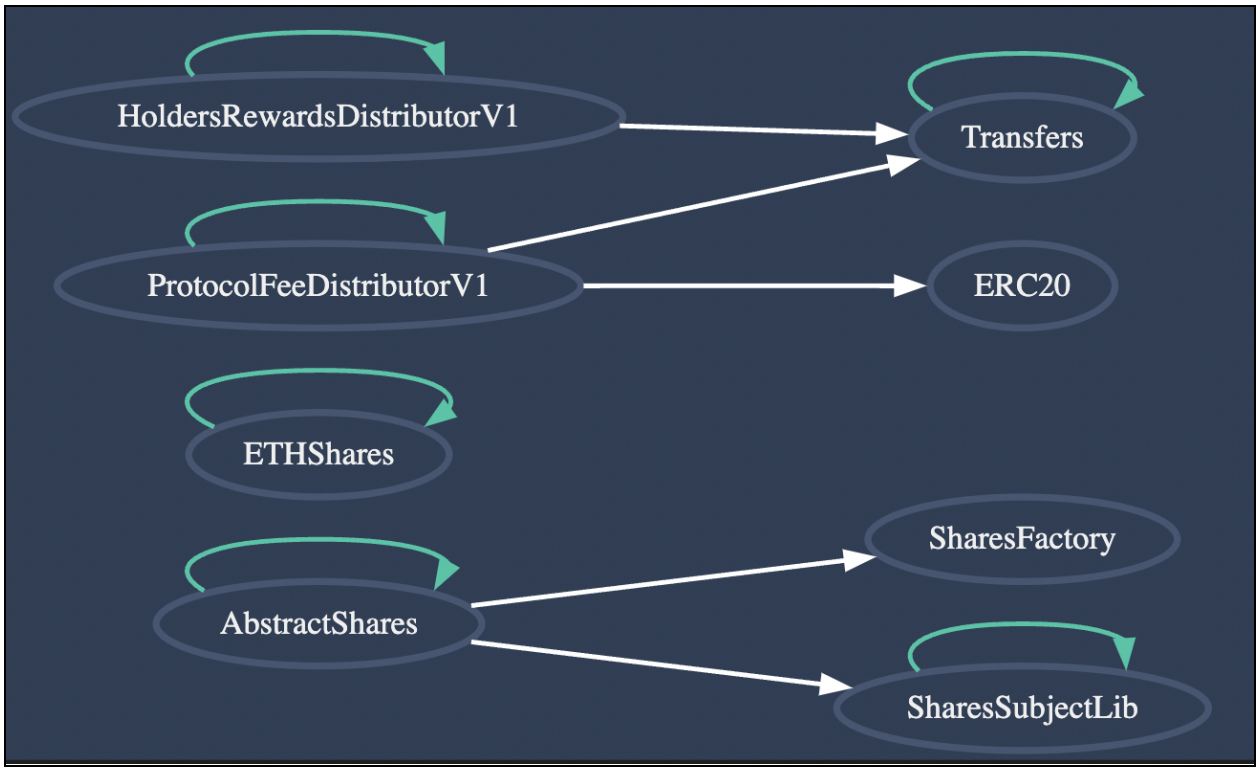
```
unchecked-transfer (1 results) (High)
uninitialized-state (2 results) (High)
locked-ether (3 results) (Medium)
uninitialized-local (1 results) (Medium)
constable-states (1 results) (Optimization)
```

Suyra Inheritance Graph



Suyra Contract Interaction Graphs





Disclaimer

As of the date of publication, the information provided in this report reflects the presently held understanding of the auditor's knowledge of security patterns as they relate to the client's contract(s), assuming that blockchain technologies, in particular, will continue to undergo frequent and ongoing development and therefore introduce unknown technical risks and flaws. The scope of the audit presented here is limited to the issues identified in the preliminary section and discussed in more detail in subsequent sections. The audit report does not address or provide opinions on any security aspects of the Solidity compiler, the tools used in the development of the contracts or the blockchain technologies themselves, or any issues not specifically addressed in this audit report.

The audit report makes no statements or warranties about the utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, the legal framework for the business model, or any other statements about the suitability of the contracts for a particular purpose, or their bug-free status.

To the full extent permissible by applicable law, the auditors disclaim all warranties, express or implied. The information in this report is provided "as is" without warranty, representation, or guarantee of any kind, including the accuracy of the information provided. The auditors hereby disclaim, and each client or user of this audit report hereby waives, releases and holds all auditors harmless from, any and all liability, damage, expense, or harm (actual, threatened, or claimed) from such use.