FYEO

# Security Assessment of evm-smart-contracts

Angel Protocol Finance

August 2023
Version 1.0

Presented by:
FYEO Inc.
PO Box 147044
Lakewood CO 80214
United States

Security Level
Public

# Table of Contents

Executive Summary ..... 2

Overview ..... 2

Key Findings ..... 2

Scope and Rules of Engagement ..... 4

Technical Analyses and Findings ..... 10

Findings ..... 11

Technical Analysis ..... 12

Conclusion ..... 12

Technical Findings ..... 13

General Observations ..... 13

Hardcoded Addresses ..... 14

Incorrect Use of Equality Operator ..... 15

Insufficient Validation of Function Arguments ..... 16

Lack of Validation of PriceFeeds ..... 18

LocalRegistrar is missing access control checks ..... 19

Oracle data may be stale or incorrect ..... 21

Test for condition which is never set ..... 22

Unchecked Return Value of ERC20.transfer ..... 23

Unchecked Return Values and Reentrancy Risk in ERC4626AP ..... 24

Unchecked return values on token operations ..... 25

Unimplemented Functions in AxelarExecutable.sol ..... 26

Unprotected Function in EndowmentMultiSigFactory ..... 27

Inconsistent ownership check ..... 29

IndexFund accepts any token ..... 30

Reentrancy Vulnerability ..... 31

Unimplemented Function _isOperator ..... 33

Balance check does not accept equality ..... 34

No Input Validation for details.duration / transactionExpiry ..... 35

Swap token does not check tokenA != tokenB ..... 36

Unchecked return value ..... 37

Use of Magic Numbers ..... 38

Code optimization ..... 39

IERC165 is duplicated ..... 42

Lack of Input Validation in FluxStrategy ..... 43

Missing Event Emits ..... 44

Missing zero address checks....45
No Input Validation in MultiSigGeneric....47
Similar functions with unclear documentation....48
The `updateFundMembers` does not remove members as might be expected....50
Unused code....51
Use of SafeMath Library with Solidity > 0.8....53
Use of Various Solidity Versions....54
Our Process....55
Methodology....55
Kickoff....55
Ramp-up....55
Review....56
Code Safety....56
Technical Specification Matching....56
Reporting....57
Verify....57
Additional Note....57
The Classification of vulnerabilities....58

## LIST OF FIGURES

Figure 1: Findings by Severity10
Figure 2: Methodology Flow55

## LIST OF TABLES

Table 1: Scope9
Table 2: Findings Overview12
Table 3: Legend for relationship graphs....11

# EXECUTIVE SUMMARY

## OVERVIEW

Angel Protocol Finance engaged FYEO Inc. to perform a Security Assessment of evm-smart-contracts.

The assessment was conducted remotely by the FYEO Security Team. Testing took place on July 11 - August 02, 2023, and focused on the following objectives:

- To provide the customer with an assessment of their overall security posture and any risks that were discovered within the environment during the engagement.
- To provide a professional opinion on the maturity, adequacy, and efficiency of the security measures that are in place.
- To identify potential issues and include improvement recommendations based on the results of our tests.

This report summarizes the engagement, tests performed, and findings. It also contains detailed descriptions of the discovered vulnerabilities, steps the FYEO Security Team took to identify and validate each issue, as well as any applicable recommendations for remediation.

## KEY FINDINGS

The following issues have been identified during the testing period. These should be prioritized for remediation to reduce the risk they pose:

- FYEO-ANGL-ID-01 – Hardcoded Addresses
- FYEO-ANGL-ID-02 – Incorrect Use of Equality Operator
- FYEO-ANGL-ID-03 – Insufficient Validation of Function Arguments
- FYEO-ANGL-ID-04 – Lack of Validation of PriceFeeds
- FYEO-ANGL-ID-05 – LocalRegistrar is missing access control checks
- FYEO-ANGL-ID-06 – Oracle data may be stale or incorrect
- FYEO-ANGL-ID-07 – Test for condition which is never set
- FYEO-ANGL-ID-08 – Unchecked Return Value of ERC20.transfer
- FYEO-ANGL-ID-09 – Unchecked Return Values and Reentrancy Risk in ERC4626AP
- FYEO-ANGL-ID-10 – Unchecked return values on token operations

- FYEO-ANGL-ID-11 – Unimplemented Functions in AxelarExecutable.sol
- FYEO-ANGL-ID-12 – Unprotected Function in EndowmentMultiSigFactory
- FYEO-ANGL-ID-13 – Inconsistent ownership check
- FYEO-ANGL-ID-14 – IndexFund accepts any token
- FYEO-ANGL-ID-15 – Reentrancy Vulnerability
- FYEO-ANGL-ID-16 – Unimplemented Function _isOperator
- FYEO-ANGL-ID-17 – Balance check does not accept equality
- FYEO-ANGL-ID-18 – No Input Validation for details.duration / transactionExpiry
- FYEO-ANGL-ID-19 – Swap token does not check tokenA != tokenB
- FYEO-ANGL-ID-20 – Unchecked return value
- FYEO-ANGL-ID-21 – Use of Magic Numbers
- FYEO-ANGL-ID-22 – Code optimization
- FYEO-ANGL-ID-23 – IERC165 is duplicated
- FYEO-ANGL-ID-24 – Lack of Input Validation in FluxStrategy
- FYEO-ANGL-ID-25 – Missing Event Emits
- FYEO-ANGL-ID-26 – Missing zero address checks
- FYEO-ANGL-ID-27 – No Input Validation in MultiSigGeneric
- FYEO-ANGL-ID-28 – Similar functions with unclear documentation
- FYEO-ANGL-ID-29 – The `updateFundMembers` does not remove members as might be expected
- FYEO-ANGL-ID-30 – Unused code
- FYEO-ANGL-ID-31 – Use of SafeMath Library with Solidity > 0.8
- FYEO-ANGL-ID-32 – Use of Various Solidity Versions

Based on our review process, we conclude that the reviewed code implements the documented functionality.

## Scope and Rules of Engagement

The FYEO Review Team performed a Security Assessment of evm-smart-contracts. The following table documents the targets in scope for the engagement. No additional systems or resources were in scope for this assessment.

The source code was supplied through a private repository at
https://github.com/AngelProtocolFinance/evm-smart-contracts with the commit hash
08ea2273ec02c013dcf6cb827ec02d7477439a26.

| Files included in the code review |
|---|

```
angelevm/
└── contracts/
    ├── accessory/
    │   └── gift-cards/
    │       ├── scripts/
    │       │   └── deploy.ts
    │       ├── GiftCards.sol
    │       ├── message.sol
    │       └── storage.sol
    ├── axelar/
    │   └── AxelarExecutable.sol
    ├── core/
    │   ├── accounts/
    │   │   ├── diamond/
    │   │   │   ├── facets/
    │   │   │   │   ├── DiamondCutFacet.sol
    │   │   │   │   ├── DiamondLoupeFacet.sol
    │   │   │   │   └── OwnershipFacet.sol
    │   │   │   ├── interfaces/
    │   │   │   │   ├── IDiamondCut.sol
    │   │   │   │   ├── IDiamondLoupe.sol
    │   │   │   │   ├── IERC165.sol
    │   │   │   │   └── IERC173.sol
    │   │   │   ├── libraries/
    │   │   │   │   └── LibDiamond.sol
    │   │   │   ├── upgradeInitializers/
    │   │   │   │   └── DiamondInit.sol
    │   │   │   └── Diamond.sol
    │   │   ├── facets/
    │   │   │   ├── AccountsAllowance.sol
    │   │   │   ├── AccountsCreateEndowment.sol
    │   │   │   ├── AccountsDaoEndowments.sol
    │   │   │   ├── AccountsDeployContract.sol
```

Files included in the code review

```
    │       │       │       ├── AccountsDepositWithdrawEndowments.sol
    │       │       │       ├── AccountsDonationMatch.sol
    │       │       │       ├── AccountsQueryEndowments.sol
    │       │       │       ├── AccountsSwapRouter.sol
    │       │       │       ├── AccountsUpdate.sol
    │       │       │       ├── AccountsUpdateEndowmentSettingsController.sol
    │       │       │       ├── AccountsUpdateEndowments.sol
    │       │       │       ├── AccountsUpdateStatusEndowments.sol
    │       │       │       ├── AccountsVaultFacet.sol
    │       │       │       └── ReentrancyGuardFacet.sol
    │       │       ├── interfaces/
    │       │       │       ├── IAccounts.sol
    │       │       │       ├── IAccountsAllowance.sol
    │       │       │       ├── IAccountsCreateEndowment.sol
    │       │       │       ├── IAccountsDaoEndowments.sol
    │       │       │       ├── IAccountsDeployContract.sol
    │       │       │       ├── IAccountsDepositWithdrawEndowments.sol
    │       │       │       ├── IAccountsDonationMatch.sol
    │       │       │       ├── IAccountsEvents.sol
    │       │       │       ├── IAccountsQueryEndowments.sol
    │       │       │       ├── IAccountsSwapRouter.sol
    │       │       │       ├── IAccountsUpdate.sol
    │       │       │       ├── IAccountsUpdateEndowmentSettingsController.sol
    │       │       │       ├── IAccountsUpdateEndowments.sol
    │       │       │       ├── IAccountsUpdateStatusEndowments.sol
    │       │       │       └── IAccountsVaultFacet.sol
    │       │       ├── lib/
    │       │       │       └── LibAccounts.sol
    │       │       ├── scripts/
    │       │       │       ├── deploy/
    │       │       │       │       ├── cutDiamond.ts
    │       │       │       │       ├── deploy.ts
    │       │       │       │       ├── deployDiamond.ts
    │       │       │       │       ├── deployFacets.ts
    │       │       │       │       ├── getFacetFactoryEntries.ts
    │       │       │       │       ├── index.ts
    │       │       │       │       └── types.ts
    │       │       │       └── libraries/
    │       │       │               └── diamond.ts
    │       │       ├── message.sol
    │       │       └── storage.sol
    │       ├── erc20ap/
    │       │       ├── ERC20AP.sol
```

| Files included in the code review |
|---|
| <pre>│       │       └── IERC20AP.sol<br>│       ├── gasFwd/<br>│       │       ├── scripts/<br>│       │       │       └── deploy.ts<br>│       │       ├── GasFwd.sol<br>│       │       ├── GasFwdFactory.sol<br>│       │       ├── IGasFwd.sol<br>│       │       └── IGasFwdFactory.sol<br>│       ├── index-fund/<br>│       │       ├── scripts/<br>│       │       │       └── deploy.ts<br>│       │       ├── IIndexFund.sol<br>│       │       ├── IndexFund.sol<br>│       │       ├── message.sol<br>│       │       └── storage.sol<br>│       ├── registrar/<br>│       │       ├── interfaces/<br>│       │       │       ├── ILocalRegistrar.sol<br>│       │       │       └── IRegistrar.sol<br>│       │       ├── lib/<br>│       │       │       └── LocalRegistrarLib.sol<br>│       │       ├── scripts/<br>│       │       │       └── deploy.ts<br>│       │       ├── LocalRegistrar.sol<br>│       │       ├── Registrar.sol<br>│       │       ├── message.sol<br>│       │       └── storage.sol<br>│       ├── router/<br>│       │       ├── scripts/<br>│       │       │       └── deploy.ts<br>│       │       ├── IRouter.sol<br>│       │       ├── Router.sol<br>│       │       └── RouterLib.sol<br>│       ├── strategy/<br>│       │       ├── APStrategy_V1.sol<br>│       │       └── IStrategy.sol<br>│       ├── vault/<br>│       │       ├── interfaces/<br>│       │       │       └── IVault.sol<br>│       │       ├── APVault_V1.sol<br>│       │       └── ERC4626AP.sol<br>│       ├── proxy.sol<br>│       └── validator.sol</pre> |

Files included in the code review

```
├── integrations/
│   ├── flux/
│   │   ├── test/
│   │   │   └── DummyFUSDC.sol
│   │   ├── FluxStrategy.sol
│   │   └── IFlux.sol
│   └── goldfinch/
│       ├── test/
│       │   ├── DummyCRVLP.sol
│       │   └── DummyStakingRewards.sol
│       ├── APGoldfinchConfig.sol
│       ├── GFITrader.sol
│       ├── GoldfinchVault.sol
│       ├── ICurveLP.sol
│       ├── IRegistrarGoldfinch.sol
│       ├── IStakingRewards.sol
│       └── StakingRewardsVesting.sol
├── lib/
│   ├── Strings/
│   │   └── string.sol
│   ├── address/
│   │   └── array.sol
│   ├── FixedPointMathLib.sol
│   ├── StringAddressUtils.sol
│   ├── array.sol
│   └── utils.sol
├── multisigs/
│   ├── interfaces/
│   │   ├── ICharityApplications.sol
│   │   └── IMultiSigGeneric.sol
│   ├── scripts/
│   │   ├── deploy.ts
│   │   ├── deployAPTeamMultiSig.ts
│   │   └── deployCharityApplications.ts
│   ├── APTeamMultiSig.sol
│   ├── CharityApplications.sol
│   ├── CharityApplicationsStorage.sol
│   ├── MultiSigGeneric.sol
│   └── storage.sol
├── normalized_endowment/
│   ├── donation-match/
│   │   ├── DonationMatch.sol
│   │   ├── DonationMatchCharity.sol
```

Files included in the code review

```
│       │       ├── DonationMatchEmitter.sol
│       │       ├── IDonationMatchEmitter.sol
│       │       ├── IDonationMatching.sol
│       │       ├── message.sol
│       │       └── storage.sol
│       ├── endowment-multisig/
│       │       ├── interfaces/
│       │       │       ├── IEndowmentMultiSigEmitter.sol
│       │       │       └── IEndowmentMultiSigFactory.sol
│       │       ├── scripts/
│       │       │       └── deploy.ts
│       │       ├── EndowmentMultiSig.sol
│       │       ├── EndowmentMultiSigEmitter.sol
│       │       └── EndowmentMultiSigFactory.sol
│       ├── incentivised-voting/
│       │       ├── interfaces/
│       │       │       ├── IIncentivisedVotingLockup.sol
│       │       │       └── QueryIIncentivisedVotingLockup.sol
│       │       ├── lib/
│       │       │       ├── shared/
│       │       │       │       ├── IBasicToken.sol
│       │       │       │       └── IERC20WithCheckpointing.sol
│       │       │       ├── Root.sol
│       │       │       └── StableMath.sol
│       │       └── IncentivisedVotingLockup.sol
│       ├── scripts/
│       │       ├── deployEmitter.ts
│       │       └── deployImplementation.ts
│       ├── subdao/
│       │       ├── Token/
│       │       │       └── ERC20.sol
│       │       ├── ISubDao.sol
│       │       ├── ISubDaoEmitter.sol
│       │       ├── SubDao.sol
│       │       ├── SubDaoEmitter.sol
│       │       ├── SubDaoLib.sol
│       │       ├── message.sol
│       │       └── storage.sol
│       └── subdao-token/
│               ├── Token/
│               │       ├── BancorBondingCurve.sol
│               │       ├── Continous.sol
│               │       └── Power.sol
```

| Files included in the code review |
|---|
| <pre>│         ├── ISubDaoToken.sol<br>│         ├── SubDaoToken.sol<br>│         ├── message.sol<br>│         └── storage.sol<br>└── test/<br>    ├── accounts/<br>    │   └── TestFacetProxyContract.sol<br>    ├── DummyERC20.sol<br>    ├── DummyGasService.sol<br>    ├── DummyGateway.sol<br>    ├── DummyRouter.sol<br>    ├── DummyStrategy.sol<br>    └── DummyVault.sol</pre> |

Table 1: Scope

# TECHNICAL ANALYSES AND FINDINGS

During the Security Assessment of evm-smart-contracts, we discovered:

- 12 findings with HIGH severity rating.
- 4 findings with MEDIUM severity rating.
- 5 findings with LOW severity rating.
- 11 findings with INFORMATIONAL severity rating.

The following chart displays the findings by severity.

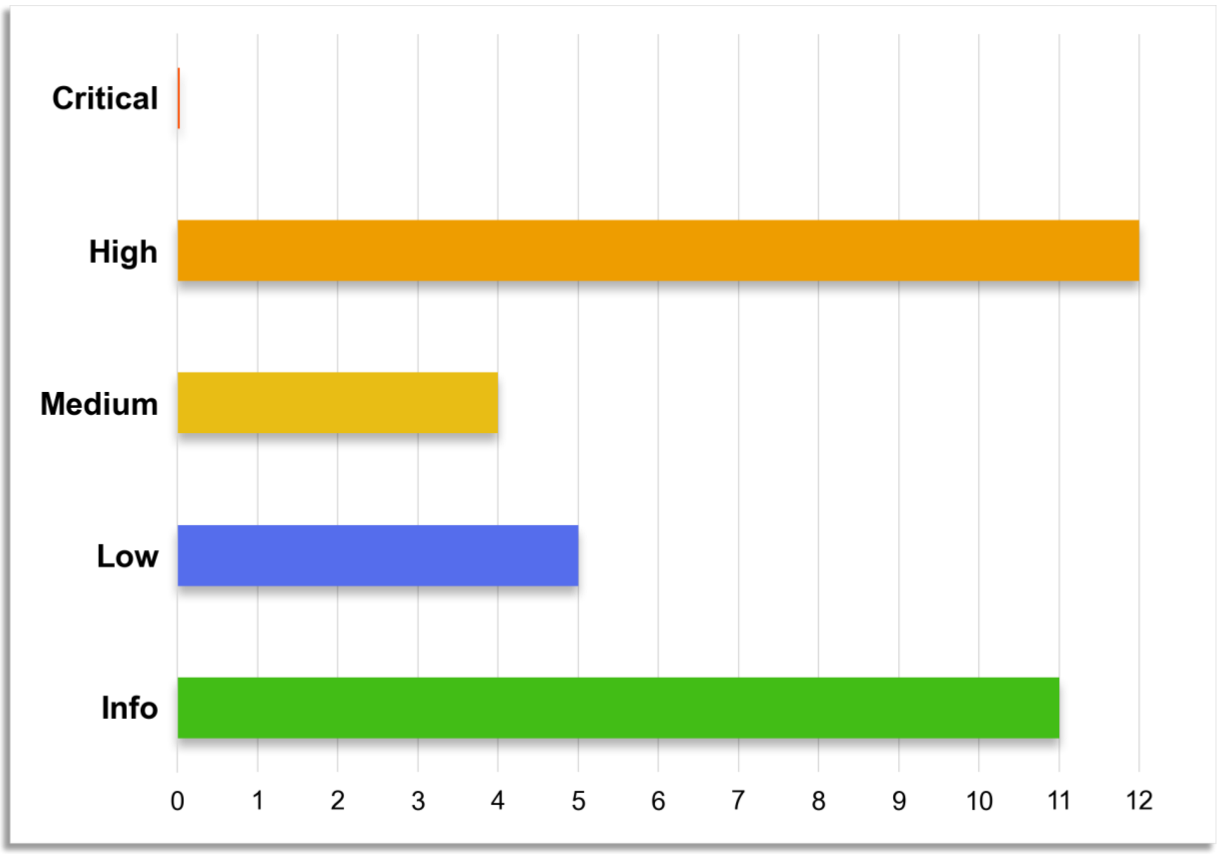


Figure 1: Findings by Severity

## FINDINGS

The *Findings* section provides detailed information on each of the findings, including methods of discovery, explanation of severity determination, recommendations, and applicable references.

The following table provides an overview of the findings.

| Finding # | Severity | Description |
|---|---|---|
| FYEO-ANGL-ID-01 | High | Hardcoded Addresses |
| FYEO-ANGL-ID-02 | High | Incorrect Use of Equality Operator |
| FYEO-ANGL-ID-03 | High | Insufficient Validation of Function Arguments |
| FYEO-ANGL-ID-04 | High | Lack of Validation of PriceFeeds |
| FYEO-ANGL-ID-05 | High | LocalRegistrar is missing access control checks |
| FYEO-ANGL-ID-06 | High | Oracle data may be stale or incorrect |
| FYEO-ANGL-ID-07 | High | Test for condition which is never set |
| FYEO-ANGL-ID-08 | High | Unchecked Return Value of ERC20.transfer |
| FYEO-ANGL-ID-09 | High | Unchecked Return Values and Reentrancy Risk in ERC4626AP |
| FYEO-ANGL-ID-10 | High | Unchecked return values on token operations |
| FYEO-ANGL-ID-11 | High | Unimplemented Functions in AxelarExecutable.sol |
| FYEO-ANGL-ID-12 | High | Unprotected Function in EndowmentMultiSigFactory |
| FYEO-ANGL-ID-13 | Medium | Inconsistent ownership check |
| FYEO-ANGL-ID-14 | Medium | IndexFund accepts any token |
| FYEO-ANGL-ID-15 | Medium | Reentrancy Vulnerability |
| FYEO-ANGL-ID-16 | Medium | Unimplemented Function _isOperator |
| FYEO-ANGL-ID-17 | Low | Balance check does not accept equality |
| FYEO-ANGL-ID-18 | Low | No Input Validation for details.duration / transactionExpiry |
| FYEO-ANGL-ID-19 | Low | Swap token does not check tokenA != tokenB |
| FYEO-ANGL-ID-20 | Low | Unchecked return value |

| | | |
|---|---|---|
| FYEO-ANGL-ID-21 | **Low** | Use of Magic Numbers |
| FYEO-ANGL-ID-22 | **Informational** | Code optimization |
| FYEO-ANGL-ID-23 | **Informational** | IERC165 is duplicated |
| FYEO-ANGL-ID-24 | **Informational** | Lack of Input Validation in FluxStrategy |
| FYEO-ANGL-ID-25 | **Informational** | Missing Event Emits |
| FYEO-ANGL-ID-26 | **Informational** | Missing zero address checks |
| FYEO-ANGL-ID-27 | **Informational** | No Input Validation in MultiSigGeneric |
| FYEO-ANGL-ID-28 | **Informational** | Similar functions with unclear documentation |
| FYEO-ANGL-ID-29 | **Informational** | The `updateFundMembers` does not remove members as might be expected |
| FYEO-ANGL-ID-30 | **Informational** | Unused code |
| FYEO-ANGL-ID-31 | **Informational** | Use of SafeMath Library with Solidity > 0.8 |
| FYEO-ANGL-ID-32 | **Informational** | Use of Various Solidity Versions |

Table 2: Findings Overview

## Technical Analysis

The source code has been manually validated to the extent that the state of the repository allowed. The validation includes confirming that the code correctly implements the intended functionality.

## Conclusion

Based on our review process, we conclude that the code implements the documented functionality to the extent of the reviewed code.

# Technical Findings

## General Observations

In our recent code review of the project, we observed a number of aspects that contribute to both its strengths and areas for improvement. The team's responsiveness to addressing issues and making adjustments has been commendable, allowing for collaborative progress. Throughout the codebase, we noticed some areas where additional comments would greatly enhance code comprehension. It could be beneficial to expand upon these comments to provide clearer insights into the functionality and purpose of various code segments. Furthermore, ensuring that comments are accurate and precise will prevent any potential confusion down the line.

In terms of code visibility, we noted instances where visibility specifiers were not declared for certain functions. While not a critical concern, explicitly specifying visibility can contribute to code clarity and help avoid ambiguity. Gas efficiency is an essential consideration in Solidity development. We observed that certain parts of the code could benefit from optimization to reduce gas consumption. This optimization could lead to more cost-effective transactions on the Ethereum network. The overall structure of the codebase is an area that could be enhanced.

The implementation currently exhibits some irregularities in its structure, which could be addressed to improve overall organization and maintainability. Documentation is crucial for maintaining a robust and sustainable codebase. Our observation indicates that the existing documentation could be enhanced to provide more comprehensive insights into the code's functionality, making it easier for both current and future developers to work with the codebase. In terms of coding styles, we noticed a mix of approaches throughout the code. Standardizing the coding style would contribute to consistency and facilitate collaboration among team members. There were instances of unfinished code present within the codebase. It would be beneficial to identify and either complete or remove these segments to ensure a clean and functional codebase. Lastly, the concept of open restructures was noted in the codebase. While restructures are a natural part of development, it's advisable to communicate and document such ongoing changes to avoid confusion among team members.

## Hardcoded Addresses

Finding ID: FYEO-ANGL-ID-01
Severity: **High**
Status: **Remediated**

### Description

The contract has hardcoded addresses for `ROUTER_ADDRESS` and `REFUND_ADDRESS`. This could potentially lead to loss of funds or other unintended behavior if these addresses are used without being properly set.

### Proof of Issue

**File name:** contracts/core/registrar/lib/LocalRegistrarLib.sol
**Line number:** 20, 21

```
  address constant ROUTER_ADDRESS = address(0);
  address constant REFUND_ADDRESS = address(0);
```

### Severity and Impact Summary

This could lead to loss of funds or other unintended behavior.

### Recommendation

Avoid hardcoding addresses in your smart contracts. Instead, consider passing these addresses as parameters to the constructor or setting them through a function call by an authorized address (like the contract owner). This way, you can change the addresses if needed. Also, make sure to add the necessary checks to prevent these addresses from being `address(0)`.

## Incorrect Use of Equality Operator

Finding ID: FYEO-ANGL-ID-02
Severity: **High**
Status: **Remediated**

### Description

In the `strategyInvest` and `strategyRedeem` functions, the contract uses the equality operator == instead of the assignment operator = when updating the `activeStrategies` mapping. This could lead to unexpected behavior as the `activeStrategies` mapping may not be updated correctly.

#### Proof of Issue

**File name:** contracts/core/accounts/facets/AccountsVaultFacet.sol
**Line number:** 116

```
state.STATES[id].balances.locked[tokenAddress] -= response.lockAmt;
state.STATES[id].balances.liquid[tokenAddress] -= response.liqAmt;
state.STATES[id].activeStrategies[strategy] == true;
```

**File name:** contracts/core/accounts/facets/AccountsVaultFacet.sol
**Line number:** 202

```
if (response.status == IVault.VaultActionStatus.POSITION_EXITED) {
  state.STATES[id].activeStrategies[strategy] == false;
}
```

**File name:** contracts/core/accounts/facets/AccountsVaultFacet.sol
**Line number:** 258

```
if (response.status == IVault.VaultActionStatus.POSITION_EXITED) {
  state.STATES[id].activeStrategies[strategy] == false;
}
```

### Severity and Impact Summary

This could lead to incorrect behavior of the contract as the `activeStrategies` mapping may not reflect the true state of the contract.

### Recommendation

Replace the equality operator == with the assignment operator = when updating the `activeStrategies` mapping.

## Insufficient Validation of Function Arguments

Finding ID: FYEO-ANGL-ID-03
Severity: **High**
Status: **Remediated**

### Description

The contract does not sufficiently validate function arguments in several places. For example, in the `depositMatic` function, it only checks that `msg.value > 0`, but does not validate the `details` argument. Similarly, in the `depositERC20` function, it only checks that `tokenAddress != address(0)`, but does not validate the `details` argument or the `amount`.

In the EVM accessing an array element that does not exist will yield an item with all 0 data. So false for bools and the first declared options for enums i.e.`LibAccounts.EndowmentType.Charity`. This function allows for execution with an invalid `details.id`. A proper check should be added.

#### Proof of Issue

**File name:** contracts/core/accounts/facets/AccountsDepositWithdrawEndowments.sol
**Line number:** 37

```
function depositMatic(AccountMessages.DepositRequest memory details) public payable
nonReentrant {
  require(msg.value > 0, "Invalid Amount");
  AccountStorage.State storage state = LibAccounts.diamondStorage();
  AccountStorage.Config memory tempConfig = state.config;
  AccountStorage.EndowmentState storage tempEndowmentState = state.STATES[details.id];
  require(!tempEndowmentState.closingEndowment, "Endowment is closed");

  ...
```

**File name:** contracts/core/accounts/facets/AccountsDepositWithdrawEndowments.sol
**Line number:** 60

```
function depositERC20(
  AccountMessages.DepositRequest memory details,
  address tokenAddress,
  uint256 amount
) public nonReentrant {
  require(tokenAddress != address(0), "Invalid Token Address");
  AccountStorage.State storage state = LibAccounts.diamondStorage();
  AccountStorage.EndowmentState storage tempEndowmentState = state.STATES[details.id];
  require(!tempEndowmentState.closingEndowment, "Endowment is closed");
  ...
```

### Severity and Impact Summary

This could lead to unexpected behavior and potential loss of funds if invalid arguments are passed to these functions.

### Recommendation

It is recommended to add validation checks for all function arguments. For example, check that the `details` argument in the `depositMatic` and `depositERC20` functions is valid, and that the `amount` in the `depositERC20` function is greater than 0.

## Lack of Validation of PriceFeeds

Finding ID: FYEO-ANGL-ID-04
Severity: **High**
Status: **Remediated**

### Description

The price feed in `updateTokenPriceFeed` is not properly checked and could be bad input. It is recommended to check the feeds data and interface compliance.

#### Proof of Issue

**File name:** contracts/core/registrar/Registrar.sol
**Line number:** 233

```
function updateTokenPriceFeed(address token, address priceFeed) public onlyOwner {
  state.PriceFeeds[token] = priceFeed;
}
```

### Severity and Impact Summary

It could lead to unexpected behavior, including loss of funds or incorrect execution of functions.

### Recommendation

Always validate function inputs. This can be done using require statements to enforce conditions on the inputs.

## LocalRegistrar is missing access control checks

Finding ID: FYEO-ANGL-ID-05
Severity: **High**
Status: **Remediated**

### Description

Changing some of the configuration is possible without authorization.

#### Proof of Issue

**File name:** contracts/core/registrar/LocalRegistrar.sol
**Line number:** 208

```
function setVaultOperatorApproved(address _operator, bool _isApproved) external
override {
  LocalRegistrarLib.LocalRegistrarStorage storage lrs =
LocalRegistrarLib.localRegistrarStorage();
  lrs.ApprovedVaultOperators[_operator] = _isApproved;
}
```

**Line number:** 213

```
function setFeeSettingsByFeesType(
  LibAccounts.FeeTypes _feeType,
  uint256 _rate,
  address _payout
) external {
  LocalRegistrarLib.LocalRegistrarStorage storage lrs =
LocalRegistrarLib.localRegistrarStorage();
  lrs.FeeSettingsByFeeType[_feeType] = LibAccounts.FeeSetting({
    payoutAddress: _payout,
    bps: _rate
  });
  emit FeeSettingsUpdated(_feeType, _rate, _payout);
}
```

**Line number:** 245

```
function setAPGoldfinchParams(
  APGoldfinchConfigLib.APGoldfinchConfig calldata _apGoldfinch
) public {
  APGoldfinchConfigLib.APGoldfinchConfig storage grs = APGoldfinchConfigLib
    .goldfinchRegistrarStorage();
  grs.crvParams.allowedSlippage = _apGoldfinch.crvParams.allowedSlippage;
}
```

### Severity and Impact Summary

Anyone can change the fees and approve operators.

### Recommendation

Make sure the authorization is implemented as required. Use the `onlyOwner` modifier as necessary.

## Oracle data may be stale or incorrect

Finding ID: FYEO-ANGL-ID-06
Severity: **High**
Status: **Remediated**

### Description

The data returned by the Chainlink oracle is not properly verified and could be stale. The number of decimals in the answer should also be considered.

#### Proof of Issue

**File name:** contracts/core/accounts/facets/AccountsSwapRouter.sol
**Line number:** 163

```
AggregatorV3Interface chainlinkFeed = AggregatorV3Interface(tokenFeed);
(, int256 answer, , , ) = chainlinkFeed.latestRoundData();
```

### Severity and Impact Summary

The price feed could be stale and provide invalid data. The number of decimals for the feed should be checked to be as expected.

### Recommendation

The `updatedAt` should be checked to be != 0 and against the `block.timestamp` allowing for some delay. The `answer` should be checked to not be 0. The `answeredInRound` should be >= `roundId`.

## TEST FOR CONDITION WHICH IS NEVER SET

Finding ID: FYEO-ANGL-ID-07
Severity: **High**
Status: **Remediated**

### Description

There are various checks against `pendingRedemptions` but this variable is never written.

### Proof of Issue

**File name:** contracts/core/accounts/facets/AccountsVaultFacet.sol
**Line number:** 165

```
require(tempEndowment.pendingRedemptions == 0, "RedemptionInProgress");
```

### Severity and Impact Summary

The code indicates missing functionality which may lead to unexpected outcomes.

### Recommendation

Make sure this functionality is implemented correctly.

## Unchecked Return Value of ERC20.transfer

Finding ID: FYEO-ANGL-ID-08
Severity: **High**
Status: **Remediated**

### Description

The contract does not check the return value of the `transfer` function. This could lead to unexpected behavior if the `transfer` function fails but the contract continues execution as if it succeeded.

### Proof of Issue

**File name:** contracts/core/gasFwd/GasFwd.sol
**Line number:** 25, 29

```
function payForGas(address token, uint256 amount) external onlyAccounts {
  IERC20(token).transfer(msg.sender, amount);
}

function sweep(address token) external onlyAccounts {
  IERC20(token).transfer(msg.sender, IERC20(token).balanceOf(address(this)));
}
```

### Severity and Impact Summary

If the `transfer` function fails but the contract continues execution as if it succeeded, it could lead to loss of funds or other unexpected behavior.

### Recommendation

Always check the return value of the `transfer` function and handle failure cases appropriately. You can use the `require` function to ensure that the `transfer` function succeeded, or use the SafeERC20 wrapper with its `safeTransfer()` function.

## Unchecked Return Values and Reentrancy Risk in ERC4626AP

Finding ID: FYEO-ANGL-ID-09
Severity: **High**
Status: **Remediated**

### Description

The contract does not check the return values of the `transferFrom` and `approve` functions. It looks like it is meant to use `safeTransfer` and `safeApprove`. While some functions are `operatorOnly` the use of reentrancy guards would help keep these calls safe.

#### Proof of Issue

**File name:** contracts/core/vault/ERC4626AP.sol
**Line number:** 57

```
asset.transferFrom(strategy, address(this), assets);
```

**Line number:** 73

```
asset.transferFrom(_msgSender(), address(this), assets);
```

**Line number:** 118

```
asset.approve(receiver, assets);
```

### Severity and Impact Summary

For certain tokens a value indicating failure may be returned in which case continuing as if the transfer succeeded could lead to loss of funds or cause unexpected behavior.

### Recommendation

Use the `safe` version of the functions and add reentrancy guards as appropriate.

## UNCHECKED RETURN VALUES ON TOKEN OPERATIONS

Finding ID: FYEO-ANGL-ID-10
Severity: **High**
Status: **Remediated**

### Description

The return value of the transfer function is not checked. A token contract may return false to indicate failure.

### Proof of Issue

**File name:** contracts/core/router/Router.sol
**Line number:** 314

```
IERC20Metadata(_action.token).transfer(
  StringToAddress.toAddress(accountsContractAddress),
  _sendAmt
);
```

**File name:** contracts/core/router/Router.sol
**Line number:** 355

```
IERC20Metadata(_action.token).transfer(apParams.refundAddr, _sendAmt);
```

**File name:** contracts/core/router/Router.sol
**Line number:** 360

```
IERC20Metadata(_action.token).transfer(apParams.refundAddr, _sendAmt);
```

**File name:** contracts/normalized_endowment/donation-match/DonationMatchCharity.sol
**Line number:** 120

```
IERC20Burnable(token).transfer(donor, donorAmount);
```

**File name:** contracts/normalized_endowment/subdao/SubDao.sol
**Line number:** 243

```
IERC20(config.daoToken).transferFrom(msg.sender, address(this), depositamount);
```

### Severity and Impact Summary

The contract could continue execution as though the transfer succeeded even though it actually failed. This could lead to loss of funds.

### Recommendation

Check the returned value or use the `safeTransfer`, `safeTransferFrom` functions.

## Unimplemented Functions in AxelarExecutable.sol

Finding ID: FYEO-ANGL-ID-11
Severity: **High**
Status: **Remediated**

### Description

The `_execute` and `_executeWithToken` functions are declared but not implemented. This could lead to unexpected behavior or vulnerabilities.

#### Proof of Issue

**File name:** contracts/axelar/AxelarExecutable.sol
**Line number:** 59

```
function _execute(
  string calldata sourceChain,
  string calldata sourceAddress,
  bytes calldata payload
) internal virtual returns (IVault.VaultActionData memory) {}

function _executeWithToken(
  string calldata sourceChain,
  string calldata sourceAddress,
  bytes calldata payload,
  string calldata tokenSymbol,
  uint256 amount
) internal virtual returns (IVault.VaultActionData memory) {}
```

### Severity and Impact Summary

This could lead to unexpected behavior or vulnerabilities.

### Recommendation

It is recommended to implement these functions or remove them if they are not needed. If these functions are meant to be overridden in a derived contract, consider declaring them as `abstract`.

## Unprotected Function in EndowmentMultiSigFactory

Finding ID: FYEO-ANGL-ID-12
Severity: **High**
Status: **Remediated**

### Description

The function `create` is public and not protected by any access control mechanism. This means that any address can call this function and create a new multisig wallet overwriting existing data.

### Proof of Issue

**File name:** contracts/normalized_endowment/endowment-multisig/EndowmentMultiSigFactory.sol
**Line number:** 74

```
function create(
  uint256 endowmentId,
  address emitterAddress,
  address[] memory owners,
  uint256 required,
  uint256 transactionExpiry
) public returns (address wallet) {
  bytes memory EndowmentData = abi.encodeWithSignature(
    "initialize(uint256,address,address[],uint256,bool,uint256)",
    endowmentId,
    emitterAddress,
    owners,
    required,
    false,
    transactionExpiry
  );
  wallet = address(new ProxyContract(IMPLEMENTATION_ADDRESS, PROXY_ADMIN,
EndowmentData));

  // >> A BETTER PLACE TO CALL THIS MIGHT BE INSIDE `EndowmentMultiSig > initialize`
FUNCTION
  IEndowmentMultiSigEmitter(emitterAddress).createMultisig(
    wallet,
    endowmentId,
    emitterAddress,
    owners,
    required,
    false,
    transactionExpiry
  );
  register(wallet);
  // also store address of multisig in endowmentIdToMultisig
  endowmentIdToMultisig[endowmentId] = wallet;
}
```

### Severity and Impact Summary

This could lead to data being overwritten.

### Recommendation

Add a check to see if `endowmentIdToMultisig[endowmentId]` is already set.

## INCONSISTENT OWNERSHIP CHECK

Finding ID: FYEO-ANGL-ID-13
Severity: **Medium**
Status: **Remediated**

### Description

Each of the 3 functions (`removeIndexFund`, `removeMember`, `updateFundMembers`) state that they can be called by rent owner. Yet each one implements a different check.

### Proof of Issue

**File name:** contracts/core/index-fund/IndexFund.sol
**Line number:** 200

```
require(msg.sender != state.config.owner, "Unauthorized");
```

**Line number:** 232

```
require(msg.sender == registrar_config.accountsContract, "Unauthorized");
```

**Line number:** 260

```
require(msg.sender == state.config.owner, "Unauthorized");
```

### Severity and Impact Summary

These statements contradict each other which could lead to unauthorized access.

### Recommendation

Implement correct authority checks.

## IndexFund accepts any token

Finding ID: FYEO-ANGL-ID-14
Severity: **Medium**
Status: **Remediated**

### Description

The IndexFunds `depositERC20` accepts any (=all) tokens. There should be a whitelist of accepted tokens.

#### Proof of Issue

**File name:** contracts/core/index-fund/IndexFund.sol
**Line number:** 291

```
function depositERC20(
  uint256 fundId,
  address token,
  uint256 amount,
  uint256 splitToLiquid
) public nonReentrant {
  require(token != address(0), "Invalid Token Address");
```

### Severity and Impact Summary

It seems that the token should be checked against the `isTokenAccepted` functionality.

### Recommendation

Make sure this was implemented as required.

## Reentrancy Vulnerability

Finding ID: FYEO-ANGL-ID-15
Severity: **Medium**
Status: **Remediated**

### Description

The Router and FluxStrategy contracts do not utilize reentrancy guards.

### Proof of Issue

While `onlyLocalAccountsContract`is currently guarding these functions, it may be appropriate to add reentrancy guards to future proof the contract and avoid any issues.

**File name:** contracts/core/router/Router.sol
**Line number:** 149

```
function _redeem(
  LocalRegistrarLib.StrategyParams memory _params,
  IVault.VaultActionData memory _action
) internal onlyOneAccount(_action) returns (IVault.VaultActionData memory) {
```

**Line number:** 199

```
function _redeemAll(
  LocalRegistrarLib.StrategyParams memory _params,
  IVault.VaultActionData memory _action
) internal onlyOneAccount(_action) returns (IVault.VaultActionData memory) {
```

**Line number:** 307

```
function _prepareAndSendTokensLocal(
  IVault.VaultActionData memory _action,
  uint256 _sendAmt
) internal returns (IVault.VaultActionData memory) {
```

**Line number:** 322

```
function _prepareAndSendTokensGMP(
  IVault.VaultActionData memory _action,
  uint256 _sendAmt
) internal returns (IVault.VaultActionData memory) {
```

**Line number:** 399

```
function _executeWithToken(
  string calldata sourceChain,
  string calldata sourceAddress,
  bytes calldata payload,
  string calldata tokenSymbol,
  // we could remove this amount as it's only used to check if liq + locked amounts
  // passed in inside payload add up to 'amount'
  // and if we need this for 'catch' statements we can just calculate it from said
values
  uint256 amount
```

```
)
  internal
  override
  onlyAccountsContract(sourceChain, sourceAddress)
  notZeroAddress(sourceAddress)
  returns (IVault.VaultActionData memory)
{
```

**File name:** contracts/integrations/flux/FluxStrategy.sol
**Line number:** 76

```
function deposit(uint256 amt) external payable whenNotPaused returns (uint256) {
```

**Line number:** 98

```
function withdraw(uint256 amt) external payable whenNotPaused returns (uint256) {
```

### Severity and Impact Summary

While these functions require an authorized sender with the `onlyAccountsContract` modifier, updates to the contract may make guards useful.

### Recommendation

Consider adding guards to functions with external calls.

## UNIMPLEMENTED FUNCTION _ISOPERATOR

Finding ID: FYEO-ANGL-ID-16
Severity: **Medium**
Status: **Remediated**

### Description

The function `_isOperator()` is declared but not implemented. This could lead to unexpected behavior.

#### Proof of Issue

**File name:** contracts/core/erc20ap/ERC20AP.sol
**Line number:** 249

```
function _isOperator(address _operator) internal virtual returns (bool) {}
```

### Severity and Impact Summary

This could lead to unexpected behavior as the function is not implemented.

### Recommendation

Implement the `_isOperator()` function or remove it if it's not needed.

## Balance check does not accept equality

Finding ID: FYEO-ANGL-ID-17
Severity: **Low**
Status: **Remediated**

### Description

Some functions that allow funds to be spent, check if the allowance is greater than the amount to be spent. However, this check should be >= instead of >.

#### Proof of Issue

**File name:** contracts/core/accounts/facets/AccountsAllowance.sol
**Line number:** 109

```
require(
  state.ALLOWANCES[endowId][msg.sender][token] > amount,
  "Amount reqested exceeds allowance balance"
);
```

**File name:** contracts/core/accounts/facets/AccountsDepositWithdrawEndowments.sol
**Line number:** 294

```
require(current_bal > tokens[t].amnt, "InsufficientFunds");
```

### Severity and Impact Summary

This could prevent users from spending their full allowance.

### Recommendation

Change the check to >= instead of >. This will allow users to spend their full allowance.

## No Input Validation for details.duration / transactionExpiry

Finding ID: FYEO-ANGL-ID-18
Severity: **Low**
Status: **Remediated**

### Description

While this setting can be modified later, various functions take the `details.duration` to set `transactionExpiry` but do not validate it. This could lead to unexpected behavior if an invalid duration is provided.

#### Proof of Issue

**File name:** contracts/core/accounts/facets/AccountsCreateEndowment.sol
**Line number:** 74

```
address owner = IEndowmentMultiSigFactory(registrar_config.multisigFactory).create(
  newEndowId,
  registrar_config.multisigEmitter,
  details.members,
  details.threshold,
  details.duration
);
```

### Severity and Impact Summary

This could lead to unexpected behavior if an invalid duration is provided.

### Recommendation

Add validation for the `details.duration` / `transactionExpiry` input. This could involve checking that the duration is within a certain range, or that it is not zero.

## SWAP TOKEN DOES NOT CHECK TOKENA != TOKENB

Finding ID: FYEO-ANGL-ID-19
Severity: **Low**
Status: **Remediated**

### Description

The `swapToken` function does not check if the two tokens are the same.

### Proof of Issue

**File name:** contracts/core/accounts/facets/AccountsSwapRouter.sol
**Line number:** 58

```
require(tokenIn != address(0) && tokenOut != address(0), "Invalid Swap Input: Zero
Address");
```

### Severity and Impact Summary

Swaps should only work between different tokens.

### Recommendation

Check that the two tokens are different.

## UNCHECKED RETURN VALUE

Finding ID: FYEO-ANGL-ID-20
Severity: **Low**
Status: **Remediated**

### Description

The call to `removeMember(uint32 member)` does not check the return value.

#### Proof of Issue

**File name:** contracts/core/accounts/facets/AccountsUpdateStatusEndowments.sol
**Line number:** 69

```
// remove closing endowment from all Index Funds that it is in
IIndexFund(registrar_config.indexFundContract).removeMember(id);
```

### Severity and Impact Summary

This could lead to unexpected behavior if the called function fails but the contract continues execution as if it succeeded.

### Recommendation

Always check the return value and handle potential failures appropriately. This can be done by wrapping the call in a `require` statement to ensure it succeeded, or by handling the failure case in an `if` statement.

## Use of Magic Numbers

Finding ID: FYEO-ANGL-ID-21
Severity: **Low**
Status: **Remediated**

### Description

The contract uses magic numbers. Magic numbers can make the code harder to understand and maintain.

### Proof of Issue

**File name:** contracts/core/accounts/facets/AccountsUpdateEndowments.sol
**Line number:** 78

```
if (details.sdgs[i] > 17 || details.sdgs[i] == 0) {
  revert("InvalidInputs");
}
```

**File name:** contracts/core/registrar/Registrar.sol
**Line number:** 50

```
collectorShare: 50,
```

### Severity and Impact Summary

This could lead to maintainability issues in the future.

### Recommendation

Consider defining these magic numbers as constants at the top of your contract with descriptive names. This will make your code easier to read and maintain.

## CODE OPTIMIZATION

Finding ID: FYEO-ANGL-ID-22
Severity: **Informational**
Status: **Remediated**

### Description

Some code optimization suggestions.

#### Proof of Issue

**File name:** contracts/core/accounts/facets/AccountsDonationMatch.sol
**Line number:** 89

```
require(msg.sender == tempEndowment.owner, "Unauthorized");

require(tempEndowment.owner != address(0), "AD E02"); //A DAO does not exist yet for
this Endowment. Please set that up first.
```

The message sender is unlikely to be the zero address, therefore the second check is redundant.

**File name:** contracts/core/vault/ERC4626AP.sol
**Line number:** 161

```
function getPricePerFullShare() public view virtual returns (uint256) {
  return
    totalSupply() == 0
      ? decimals()
      : (asset.balanceOf(address(this)) * decimals()) / totalSupply();
}
```

The function `totalSupply()` is called twice.

**File name:** contracts/multisigs/MultiSigGeneric.sol
**Line number:** 125

```
function addOwners(address[] memory owners) public virtual override onlyWallet {
  require(owners.length > 0, "Empty new owners list passed");
  for (uint256 o = 0; o < owners.length; o++) {
    require(!isOwner[owners[o]], "New owner already exists");
    // increment active owners count by 1
    activeOwnersCount += 1;
    // set the owner address to false in mapping
    isOwner[owners[o]] = true;
    emit OwnerAdded(owners[o]);
  }
}
```

Since it is required that every member is new, the `activeOwnersCount` can be updated once after the loop instead of on each iteration.

**File name:** contracts/multisigs/MultiSigGeneric.sol
**Line number:** 148

```
activeOwnersCount -= 1;
```

Same as above but in `removeOwners`.

**File name:** contracts/normalized_endowment/donation-match/DonationMatch.sol
**Line number:** 30

```
contract DonationMatch is Storage, Initializable
```

This looks as though it implements `IDonationMatching`.

**File name:** contracts/normalized_endowment/subdao/ISubDao.sol
**Line number:** 45

```
function buildDaoTokenMesage(SubDaoMessages.InstantiateMsg memory msg) external;
```

The variable `msg` shadows the `msg` type.

**File name:** contracts/normalized_endowment/subdao-token/SubDaoToken.sol
**Line number:** 78

```
uint256 donorAmount = (totalMinted.mul(40)).div(100);
uint256 endowmentAmount = (totalMinted.mul(40)).div(100);
```

These will yield the same result.

**File name:** contracts/core/accounts/facets/AccountsDepositWithdrawEndowments.sol
**Line number:** 267

```
IRegistrar(state.config.registrarContract)
  .getFeeSettingsByFeeType(LibAccounts.FeeTypes.EarlyLockedWithdrawCharity)
  .bps
```

Is called in a loop but should be called once.

**File name:** contracts/core/accounts/facets/AccountsDepositWithdrawEndowments.sol
**Line number:** 277

```
withdrawFeeRateAp = IRegistrar(state.config.registrarContract)
  .getFeeSettingsByFeeType(LibAccounts.FeeTypes.WithdrawCharity)
  .bps
```

Is called in a loop but should be called once.

**File name:** contracts/core/accounts/facets/AccountsDepositWithdrawEndowments.sol
**Line number:** 281

```
withdrawFeeRateAp = IRegistrar(state.config.registrarContract)
  .getFeeSettingsByFeeType(LibAccounts.FeeTypes.WithdrawNormal)
  .bps;
```

Is called in a loop but should be called once.

**File name:** contracts/core/vault/APVault_V1.sol
**Line number:** 191

```
LibAccounts.FeeSetting memory feeSetting = IRegistrar(vaultConfig.registrar)
  .getFeeSettingsByFeeType(LibAccounts.FeeTypes.Harvest);
```

Is called in a loop but should be called once.

**File name:** contracts/core/vault/ERC4626AP.sol
**Line number:** 38

```
constructor(
  IERC20Metadata _asset,
  string memory _name,
  string memory _symbol
) ERC20AP(_name, _symbol, _asset.decimals()) {
  asset = _asset;
}
```

The _name and _symbol variables shadow declarations in ERC20AP.

**File name:** contracts/core/accounts/facets/AccountsUpdateEndowmentSettingsController.sol
**Line number:** 37

```
function updateEndowmentSettings(
  AccountMessages.UpdateEndowmentSettingsRequest memory details
) public nonReentrant {
  ...
```

While emitting events is of course great, emitting one event per updated setting my not be ideal.

**File name:** contracts/core/router/Router.sol
**Line number:** 335

```
uint256 liqGas = (gasFee * ((_action.liqAmt * PRECISION) / _sendAmt)) / PRECISION;
```

The division is done before the multiplication and could lead to loss of accuracy.

### Severity and Impact Summary

Not a security concern.

### Recommendation

Recommendations to improve the code and or reduce gas consumption.

## IERC165 IS DUPLICATED

Finding ID: FYEO-ANGL-ID-23
Severity: **Informational**
Status: **Remediated**

### Description

The code for the IERC165 is duplicated as it is already present in the openzeppelin library.

#### Proof of Issue

**File name:** contracts/core/accounts/diamond/interfaces/IERC165.sol
**Line number:** 1-12

**File name:** node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol

### Severity and Impact Summary

No security impact.

### Recommendation

Use the imported code.

## Lack of Input Validation in FluxStrategy

Finding ID: FYEO-ANGL-ID-24
Severity: **Informational**
Status: **Remediated**

### Description

The contract does not validate the input for the `amt` parameter in the `deposit` and `withdraw` functions. This can lead to unexpected behavior if the input is not as expected.

#### Proof of Issue

**File name:** contracts/integrations/flux/FluxStrategy.sol
**Line number:** 76

```
function deposit(uint256 amt) external payable whenNotPaused returns (uint256) {
  if (!IFlux(config.baseToken).transferFrom(_msgSender(), address(this), amt)) {
    revert TransferFailed();
  }
  if (!IFlux(config.baseToken).approve(config.yieldToken, amt)) {
    revert ApproveFailed();
  }
  uint256 yieldTokens = _enterPosition(amt);
  if (!IFlux(config.yieldToken).approve(_msgSender(), yieldTokens)) {
    revert ApproveFailed();
  }
  emit EnteredPosition(amt, yieldTokens);
  return yieldTokens;
}
```

### Severity and Impact Summary

This can lead to unexpected behavior in the contract, potentially wasting users funds in gas fees while doing nothing.

### Recommendation

Always validate function inputs. Require that the `amt` is greater than 0.

## Missing Event Emits

Finding ID: FYEO-ANGL-ID-25
Severity: **Informational**
Status: **Remediated**

### Description

The contract has commented out `emit` statements. This could make it difficult to track state changes in the contract as events are not emitted.

#### Proof of Issue

**File name:** contracts/core/accounts/facets/AccountsVaultFacet.sol
**Line number:** 117, 200, 256

```
// emit EndowmentStateUpdated(id);
...
// emit EndowmentStateUpdated(id);
...
// emit EndowmentStateUpdated(id);
```

### Severity and Impact Summary

This could make it difficult to track state changes in the contract.

### Recommendation

Make sure the code emits events as necessary.

## MISSING ZERO ADDRESS CHECKS

Finding ID: FYEO-ANGL-ID-26
Severity: **Informational**
Status: **Remediated**

### Description

While many functions implement checks against the zero address, not all of them do. This could lead to accidental loss of contract ownership if a zero address is passed.

#### Proof of Issue

**File name:** contracts/core/accounts/diamond/facets/OwnershipFacet.sol
**Line number:** 8

```
function transferOwnership(address newOwner) external override {
  LibDiamond.enforceIsContractOwner();
  LibDiamond.setContractOwner(newOwner);
}
```

**File name:** contracts/core/gasFwd/GasFwd.sol
**Line number:** 13

```
function initialize(address _accounts) public initializer {
  accounts = _accounts;
}
```

**File name:** contracts/integrations/goldfinch/GFITrader.sol
**Line number:** 17

```
constructor(address _swapRouterAddr, address _gfi, address _weth9, address _usdc) {
  swapRouter = ISwapRouter(_swapRouterAddr);
  GFI = _gfi;
  WETH9 = _weth9;
  USDC = _usdc;
}
```

**File name:** contracts/normalized_endowment/endowment-multisig/EndowmentMultiSigFactory.sol
**Line number:** 54

```
function updateImplementation(address implementationAddress) public onlyOwner {
  IMPLEMENTATION_ADDRESS = implementationAddress;
  emit ImplementationUpdated(implementationAddress);
}
```

**File name:** contracts/normalized_endowment/endowment-multisig/EndowmentMultiSigFactory.sol
**Line number:** 62

```
function updateProxyAdmin(address proxyAdmin) public onlyOwner {
  PROXY_ADMIN = proxyAdmin;
  emit ProxyAdminUpdated(proxyAdmin);
}
```

**File name:** contracts/core/accounts/facets/AccountsDonationMatch.sol
**Line number:** 57

```
function withdrawDonationMatchErC20(uint32 id, address recipient, uint256 amount)
public {
  ...

  require(IERC20(tempEndowment.daoToken).transfer(recipient, amount), "Transfer
failed");
  emit DonationWithdrawn(id, recipient, tempEndowment.daoToken, amount);
}
```

**File name:** contracts/core/accounts/diamond/Diamond.sol
**Line number:** 15

```
constructor(address contractowner, address diamondcutfacet) payable {
  LibDiamond.setContractOwner(contractowner);

  // Add the diamondCut external function from the diamondCutFacet
  IDiamondCut.FacetCut[] memory cut = new IDiamondCut.FacetCut[](1);
  bytes4[] memory functionSelectors = new bytes4[](1);
  functionSelectors[0] = IDiamondCut.diamondCut.selector;
  cut[0] = IDiamondCut.FacetCut({
    facetAddress: diamondcutfacet,
    action: IDiamondCut.FacetCutAction.Add,
    functionSelectors: functionSelectors
  });
  LibDiamond.diamondCut(cut, address(0), "");
}
```

### Severity and Impact Summary

Checking for the zero address could prevent accidents and loss of control.

### Recommendation

Add a require statement to check against the zero address.

## No Input Validation in MultiSigGeneric

Finding ID: FYEO-ANGL-ID-27
Severity: **Informational**
Status: **Remediated**

### Description

The contract does not validate the inputs of the `initialize` function. This could potentially lead to unexpected behavior if invalid inputs are provided. For instance, while it is checked that `require(owners.length > 0, "Must pass at least one owner address")`, there is no check that `_approvalsRequired` is a sensible value.

### Proof of Issue

**File name:** contracts/multisigs/MultiSigGeneric.sol
**Line number:** 106

```
require(owners.length > 0, "Must pass at least one owner address");
for (uint256 i = 0; i < owners.length; i++) {
  require(!isOwner[owners[i]] && owners[i] != address(0));
  isOwner[owners[i]] = true;
  emit OwnerAdded(owners[i]);
}
// set storage variables
approvalsRequired = _approvalsRequired;
emit ApprovalsRequiredChanged(_approvalsRequired);
```

### Severity and Impact Summary

The contract could behave unexpectedly if invalid inputs are provided.

### Recommendation

Consider adding require statements to validate the inputs of the function. For example, you could check that `_required` is greater than 0 and less than or equal to the number of `_owners`, and that `_transactionExpiry` is a reasonable number.

## SIMILAR FUNCTIONS WITH UNCLEAR DOCUMENTATION

Finding ID: FYEO-ANGL-ID-28
Severity: **Informational**
Status: **Remediated**

### Description

These two functions are rather similar. One updates the claimed state, the other does not. The documentation on these does not seem to clearly indicate what is happening.

### Proof of Issue

**File name:** contracts/normalized_endowment/subdao-token/SubDaoToken.sol
**Line number:** 183

```
/**
 * @notice This function is used to check if the release time of the token has passed
and if the token can be claimed.
 * @param sender address
 * @return amount Amount of claimable tokens
 */
function claimTokensCheck(address sender) internal returns (uint256) {
  uint256 amount = 0;

  for (uint256 i = 0; i < CLAIM_AMOUNT[sender].details.length; i++) {
    if (
      CLAIM_AMOUNT[sender].details[i].releaseTime < block.timestamp &&
      !CLAIM_AMOUNT[sender].details[i].isClaimed
    ) {
      amount += CLAIM_AMOUNT[sender].details[i].amount;
      CLAIM_AMOUNT[sender].details[i].isClaimed = true;
      // ISubDaoTokenEmitter(emitterAddress).claimSt(sender, i);
    }
  }

  return amount;
}

/**
 * @notice This function is used to check if the token can be claimed.
 * @return amount Amount of claimable tokens
 */
function checkClaimableTokens() public view returns (uint256) {
  uint256 amount = 0;

  for (uint256 i = 0; i < CLAIM_AMOUNT[msg.sender].details.length; i++) {
    if (
      CLAIM_AMOUNT[msg.sender].details[i].releaseTime < block.timestamp &&
      !CLAIM_AMOUNT[msg.sender].details[i].isClaimed
    ) {
      amount += CLAIM_AMOUNT[msg.sender].details[i].amount;
    }
  }
  return amount;
}
```

### Severity and Impact Summary

This code is confusing and could lead to maintenance issues.

### Recommendation

Consider if having one function that takes a bool isn’t better in this case.

## THE `UPDATEFUNDMEMBERS` DOES NOT REMOVE MEMBERS AS MIGHT BE EXPECTED

Finding ID: FYEO-ANGL-ID-29
Severity: **Informational**
Status: **Remediated**

### Description

The `updateFundMembers` can be used to remove members but does not do the inverse of adding members.

### Proof of Issue

**File name:** contracts/core/index-fund/IndexFund.sol
**Line number:** 268

```
bool found;
uint256 index;
for (uint i = 0; i < members.length; i++) {
  uint256[] memory funds = state.FUNDS_BY_ENDOWMENT[members[i]];
  (index, found) = Array.indexOf(funds, fundId);
  if (!found) {
    state.FUNDS_BY_ENDOWMENT[members[i]].push(fundId);
  }
}
```

### Severity and Impact Summary

While the function correctly adds new funds to each member's list when they are added, it does not handle the scenario of removing a fund member. If a member is removed from the members array, their association with the fund will not be removed from the `FUNDS_BY_ENDOWMENT` mapping.

### Recommendation

Check what the expected outcome of removing a member is and implement the functionality as required.

## UNUSED CODE

Finding ID: FYEO-ANGL-ID-30
Severity: **Informational**
Status: **Remediated**

### Description

There is unused code throughout the project.

### Proof of Issue

**File name:** contracts/core/accounts/lib/LibAccounts.sol
**Line number:** 111

```
bytes4 constant InterfaceId_Invalid = 0xffffffff;
...
bytes4 constant InterfaceId_ERC721 = 0x80ac58cd;
```

**File name:** contracts/integrations/goldfinch/StakingRewardsVesting.sol
**Line number:** 12

```
uint256 internal constant PERCENTAGE_DECIMALS = 1e18;
```

**File name:** contracts/integrations/goldfinch/APGoldfinchConfig.sol
**Line number:** 6

```
uint256 constant DEFAULT_SLIPPAGE = 1; // 1%
```

**File name:** contracts/core/strategy/APStrategy_V1.sol
**Line number:** 78

```
function _beforeDeposit(uint256 amt) internal virtual {}

function _afterDeposit(uint256 amt) internal virtual {}

function _beforeWithdraw(uint256 amt) internal virtual {}

function _afterWithdraw(uint256 amt) internal virtual {}
```

**File name:** contracts/integrations/goldfinch/StakingRewardsVesting.sol
**Line number:** 29

```
function claim(Rewards storage rewards, uint256 reward) internal {
  rewards.totalClaimed = rewards.totalClaimed.add(reward);
}

function claimable(Rewards storage rewards) internal view returns (uint256) {
  return
rewards.totalVested.add(rewards.totalPreviouslyVested).sub(rewards.totalClaimed);
}

function checkpoint(Rewards storage rewards) internal {
  uint256 newTotalVested = totalVestedAt(
    rewards.startTime,
    rewards.endTime,
    block.timestamp,
    rewards.rentGrant()
  );

  if (newTotalVested > rewards.totalVested) {
    uint256 difference = newTotalVested.sub(rewards.totalVested);
    rewards.totalUnvested = rewards.totalUnvested.sub(difference);
    rewards.totalVested = newTotalVested;
  }
}
```

**File name:** contracts/normalized_endowment/subdao/Token/ERC20.sol
**File name:** contracts/normalized_endowment/subdao-token/Token/Continous.sol
**Line number:** 40, 120

```
function _msgData()
```

**File name:** contracts/core/accounts/facets/AccountsSwapRouter.sol
**Line number:** 174

```
function sortTokens(
  address tokenA,
  address tokenB
) internal pure returns (address token0, address token1) {
  require(tokenA != tokenB, "UniswapV3Library: IDENTICAL_ADDRESSES");
  (token0, token1) = tokenA < tokenB ? (tokenA, tokenB) : (tokenB, tokenA);
  require(token0 != address(0), "UniswapV3Library: ZERO_ADDRESS");
}
```

### Severity and Impact Summary

No security impact. Unused code however hints at missing functionality.

### Recommendation

Implement or remove as required.

## Use of SafeMath Library with Solidity > 0.8

Finding ID: FYEO-ANGL-ID-31
Severity: **Informational**
Status: **Remediated**

### Description

The contract uses the `SafeMath` library for arithmetic operations to prevent integer overflow and underflow attacks. However, since Solidity 0.8.0, the compiler automatically inserts these checks, so the use of `SafeMath` is no longer necessary.

### Proof of Issue

```
using SafeMath for uint256;
```

**File name:** contracts/core/accounts/facets/AccountsDepositWithdrawEndowments.sol
**Line number:** 31

**File name:** contracts/core/accounts/facets/AccountsSwapRouter.sol
**Line number:** 24

**File name:** contracts/core/index-fund/IndexFund.sol
**Line number:** 43

**File name:** contracts/integrations/goldfinch/StakingRewardsVesting.sol
**Line number:** 9

**File name:** contracts/normalized_endowment/subdao-token/SubDaoToken.sol
**Line number:** 23

**File name:** contracts/normalized_endowment/subdao-token/Token/BancorBondingCurve.sol
**Line number:** 15

**File name:** contracts/normalized_endowment/subdao-token/Token/Continous.sol
**Line number:** 16

### Severity and Impact Summary

No security impact. Redundant code and possibly increased gas costs.

### Recommendation

Remove the `SafeMath` library and rely on the built-in overflow and underflow checks provided by the Solidity compiler (version 0.8.0 and later).

## Use of Various Solidity Versions

Finding ID: FYEO-ANGL-ID-32
Severity: **Informational**
Status: **Remediated**

### Description

The contracts use various Solidity versions. It's recommended to always use the latest stable version of Solidity to benefit from the latest features, optimizations, and bug fixes.

#### Proof of Issue

Each file declares a compiler version.

### Severity and Impact Summary

Could potentially miss out on important updates, optimizations, and bug fixes.

### Recommendation

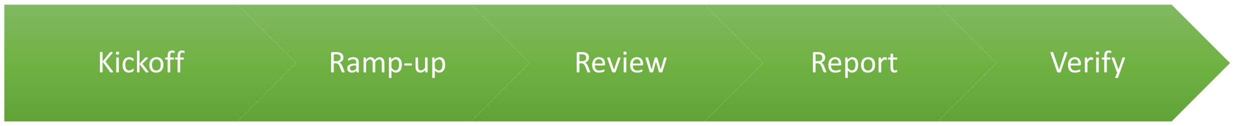
Upgrade to the latest stable version of Solidity.

# OUR PROCESS

## METHODOLOGY

FYEO Inc. uses the following high-level methodology when approaching engagements. They are broken up into the following phases.

Figure 2: Methodology Flow



### KICKOFF

The project is kicked off as the sales process has concluded. We typically set up a kickoff meeting where project stakeholders are gathered to discuss the project as well as the responsibilities of participants. During this meeting we verify the scope of the engagement and discuss the project activities. It's an opportunity for both sides to ask questions and get to know each other. By the end of the kickoff there is an understanding of the following:

- Designated points of contact
- Communication methods and frequency
- Shared documentation
- Code and/or any other artifacts necessary for project success
- Follow-up meeting schedule, such as a technical walkthrough
- Understanding of timeline and duration

### RAMP-UP

Ramp-up consists of the activities necessary to gain proficiency on the project. This can include the steps needed for familiarity with the codebase or technological innovation utilized. This may include, but is not limited to:

- Reviewing previous work in the area including academic papers
- Reviewing programming language constructs for specific languages
- Researching common flaws and recent technological advancements

## Review

The review phase is where most of the work on the engagement is completed. This is the phase where we analyze the project for flaws and issues that impact the security posture. Depending on the project this may include an analysis of the architecture, a review of the code, and a specification matching to match the architecture to the implemented code.

In this code audit, we performed the following tasks:

1. Security analysis and architecture review of the original protocol
2. Review of the code written for the project
3. Compliance of the code with the provided technical documentation

The review for this project was performed using manual methods and utilizing the experience of the reviewer. No dynamic testing was performed, only the use of custom-built scripts and tools were used to assist the reviewer during the testing. We discuss our methodology in more detail in the following sections.

## Code Safety

We analyzed the provided code, checking for issues related to the following categories:

- General code safety and susceptibility to known issues
- Poor coding practices and unsafe behavior
- Leakage of secrets or other sensitive data through memory mismanagement
- Susceptibility to misuse and system errors
- Error management and logging

This list is general and not comprehensive, meant only to give an understanding of the issues we are looking for.

## Technical Specification Matching

We analyzed the provided documentation and checked that the code matches the specification. We checked for things such as:

- Proper implementation of the documented protocol phases
- Proper error handling
- Adherence to the protocol logical description

### Reporting

FYEO Inc. delivers a draft report that contains an executive summary, technical details, and observations about the project.

The executive summary contains an overview of the engagement including the number of findings as well as a statement about our general risk assessment of the project. We may conclude that the overall risk is low but depending on what was assessed we may conclude that more scrutiny of the project is needed.

We report security issues identified, as well as informational findings for improvement, categorized by the following labels:

- Critical
- High
- Medium
- Low
- Informational

The technical details are aimed more at developers, describing the issues, the severity ranking and recommendations for mitigation.

As we perform the audit, we may identify issues that aren't security related, but are general best practices and steps that can be taken to lower the attack surface of the project. We will call those out as we encounter them and as time permits.

As an optional step, we can agree on the creation of a public report that can be shared and distributed with a larger audience.

### Verify

After the preliminary findings have been delivered, this could be in the form of the approved communication channel or delivery of the draft report, we will verify any fixes within a window of time specified in the project. After the fixes have been verified, we will change the status of the finding in the report from open to remediated.

The output of this phase will be a final report with any mitigated findings noted.

## Additional Note

It is important to note that, although we did our best in our analysis, no code audit or assessment is a guarantee of the absence of flaws. Our effort was constrained by resource and time limits along with the scope of the agreement.

While assessing the severity of the findings, we considered the impact, ease of exploitability, and the probability of attack. This is a solid baseline for severity determination.

## THE CLASSIFICATION OF VULNERABILITIES

Security vulnerabilities and areas for improvement are weighted into one of several categories using, but is not limited to, the criteria listed below:

Critical – vulnerability will lead to a loss of protected assets

- This is a vulnerability that would lead to immediate loss of protected assets
- The complexity to exploit is low
- The probability of exploit is high

High - vulnerability has potential to lead to a loss of protected assets

- All discrepancies found where there is a security claim made in the documentation that cannot be found in the code
- All mismatches from the stated and actual functionality
- Unprotected key material
- Weak encryption of keys
- Badly generated key materials
- Txn signatures not verified
- Spending of funds through logic errors
- Calculation errors overflows and underflows

Medium - vulnerability hampers the uptime of the system or can lead to other problems

- Insecure calls to third party libraries
- Use of untested or nonstandard or non-peer-reviewed crypto functions
- Program crashes, leaves core dumps or writes sensitive data to log files

Low – vulnerability has a security impact but does not directly affect the protected assets

- Overly complex functions
- Unchecked return values from 3rd party libraries that could alter the execution flow

Informational

- General recommendations